NS1 Python SDK Documentation

Release 0.18.0

NSONE, Inc.

Aug 22, 2023

Contents

NS1.

CHAPTER 1

About

This package provides an SDK for accessing the NS1 DNS platform and includes both a simple NS1 REST API wrapper as well as a higher level interface for managing zones, records, data feeds, and more. It supports synchronous and asynchronous transports.

Both python 2.7 and 3.3 are supported.

Install with:

```
$ pip install ns1-python
```

CHAPTER 2

Quick Start

First, you'll need an API Key. To create one, login to the portal and click on the Account button in the top right. Select Settings & Users, then add a new API Key at the bottom.

Simple example:

```
from ns1 import NS1

api = NS1(apiKey='<<CLEARTEXT API KEY>>')
zone = api.createZone('example.com', nx_ttl=3600)
print(zone)
record = zone.add_A('honey', ['1.2.3.4', '5.6.7.8'])
print(record)
```

Note that all zone and record changes propagate in real time throughout the NS1 platform.

There are more examples in the examples directory.

CHAPTER 3

Contributions

We welcome contributions! Please fork on GitHub and submit a Pull Request.

CHAPTER 4

Contents

4.1 Features

- Extensive config system with support for multiple API keys
- High level interface to Zones and Records
- Low level REST wrapper for all other functionality
- Extendable transport system with synchronous and asynchronous transports

4.2 Configuration

Configuring the SDK can be done programmatically and/or via loading (and saving) simple JSON text configuration files. At a minimum, the NS1 API key to access the REST API must be specified.

4.2.1 Loading From a File

By default, configuration is loaded from the file `~/.nsone`; that is, a file called `.nsone` in the home directory of the user calling the script.

```
# to load an explicit configuration file:
api = NS1(configFile='/etc/ns1/api.json')
```

4.2.2 From an API Key

```
# to generate a configuration based on an api key
api = NS1(apiKey='<<CLEARTEXT API KEY>>')
```

4.2.3 JSON File Format

This example shows two different API keys. Which to use can be selected at runtime, see *ns1.config*

```
{
   "default_key": "account2",
   "verbosity": 5,
   "keys": {
        "account1": {
            "key": "<<CLEARTEXT API KEY>>",
            "desc": "account number 1",
            "writeLock": true
        },
        "account2": {
            "key": "<<ANOTHER CLEARTEXT API KEY>>",
            "desc": "account number 2",
            "writeLock": false
        },
   },
   "cli": {
       "output_format": "text"
   }
}
```

4.2.4 More

There are more examples in the config.py example. For the full Config object reference API, see *ns1.config*

4.3 Usage

There are many examples of usage in the examples directory.

CHAPTER 5

Reference

5.1 ns1 package

5.1.1 ns1.NS1

This top level object is used to initialize and coordinate access to the NS1 platform. With it, you create objects for accessing either the basic REST interface, or the high level objects such as Zone and Record.

class ns1.**NS1**(*apiKey=None*, *config=None*, *configFile=None*, *keyID=None*)

Create a new top level NS1 API object

Parameters

- **apiKey** (*str*) – if given, initialize config with this API key (obtainable via creation in NS1 portal)
- **config** (ns1.config.Config) – if given, uses a separately constructed and configured Config object
- **configFile** (*str*) – if given, load configuration from the given json configuration file
- **keyID** (*str*) – if given, use the specified key config in the multi-key configuration file

acls()

Return a new raw REST interface to ACL resources

Return type ns1.rest.acls.Acls

addresses()

Return a new raw REST interface to address resources

Return type ns1.rest.ipam.Adresses

apikey()

Return a new raw REST interface to API key resources

Return type *ns1.rest.apikey.APIKey*

client_classes()
Return a new raw REST interface to Client Classes resources

Return type ns1.rest.client_classes.ClientClasses

createAddress(*prefix*, *status*, *network_id*, *callback=None*, *errback=None*, ***kwargs*)
Create a new Address For the list of keywords available, see *ns1.rest.ipam.Addresses.INT_FIELDS* and *ns1.rest.ipam.Addresses.PASSTHRU_FIELDS*

Parameters

- **prefix** (*str*) – CIDR prefix of the address to be created
- **status** (*str*) – The status of address assignment (planned or assigned)
- **network_id** (*int*) – network_id associated with the address

createMonitor(*callback=None*, *errback=None*, ***kwargs*)
Create a monitor

createNetwork(*name*, *scope_group_id=None*, *callback=None*, *errback=None*, ***kwargs*)
Create a new Network For the list of keywords available, see *ns1.rest.ipam.Networks.INT_FIELDS* and *ns1.rest.ipam.Networks.PASSTHRU_FIELDS*

Parameters

- **name** (*str*) – Name of the Network to be created
- **scope_group** (*int*) – (Optional) id of an existing scope group to associate with

createReservation(*scopegroup_id*, *address_id*, *mac*, *dhcp_options=None*, *callback=None*, *errback=None*, ***kwargs*)
Create a new Reservation For the list of keywords available, see ns1.rest.ipam.Reservation.INT_FIELDS and ns1.rest.ipam.Reservation.PASSTHRU_FIELDS

Parameters

- **scopegroup_id** (*int*) – id of the scope group
- **address_id** (*int*) – id of the address the reservation is associated with
- **mac** (*str*) – mac address of the reservation
- **options** (*list*) – dhcp options of the reservation

createScope(*scopegroup_id*, *address_id*, *dhcp_options=None*, *callback=None*, *errback=None*, ***kwargs*)
Create a new Scope For the list of keywords available, see ns1.rest.ipam.Scope.INT_FIELDS and ns1.rest.ipam.Scope.PASSTHRU_FIELDS

Parameters

- **scopegroup_id** (*int*) – id of the scope group
- **address_id** (*int*) – id of the address the scope is associated with
- **options** (DHCPOptions) – DHCPOptions object that contains the settings for the scope

createScopeGroup(*name*, *service_def_id*, *dhcp4*, *dhcp6*, *callback=None*, *errback=None*, ***kwargs*)
Create a new Scope Group For the list of keywords available, see ns1.rest.ipam.ScopeGroups.INT_FIELDS and ns1.rest.ipam.ScopeGroups.PASSTHRU_FIELDS

Parameters

- **name** (*str*) – Name of the Scope Group to be created

- **service_group_id** (*int*) – id of the service group the Scope group is associated with
- **dhcp4** (*ns1.ipam.DHCPIOptions*) – DHCPOptions object that contains the options for dhcpv4
- **dhcp6** (*ns1.ipam.DHCPIOptions*) – DHCPOptions object that contains the options for dhcpv6

createZone(*zone*, *zoneFile=None*, *callback=None*, *errback=None*, ***kwargs*)
Create a new zone, and return an associated high level Zone object. Several optional keyword arguments are available to configure the SOA record.

If zoneFile is specified, upload the specific zone definition file to populate the zone with.

Parameters

- **zone** (*str*) – zone name, like 'example.com'
- **zoneFile** (*str*) – absolute path of a zone file
- **retry** (*int*) – retry time
- **refresh** (*int*) – refresh ttl
- **expiry** (*int*) – expiry ttl
- **nx_ttl** (*int*) – nxdomain TTL

Return type *ns1.zones.Zone*

datafeed()
Return a new raw REST interface to feed resources

Return type *ns1.rest.data.Feed*

datasource()
Return a new raw REST interface to datasource resources

Return type *ns1.rest.data.Source*

dhcp_option_spaces()
Return a new raw REST interface to DHCP Option Spaces resources

Return type ns1.rest.dhcp_option_spaces.DHCOptionSpaces

generateDHCPOptionsTemplate(*address_family*)
Generate boilerplate dictionary to hold dhcp options

Parameters **address_family** (*str*) – dhcpv4 or dhcpv6

Returns dict containing valid option set for address family

loadAddressbyID(*id*, *callback=None*, *errback=None*)
Load an existing address by ID into a high level Address object

Parameters **id** (*int*) – id of an existing Address

loadAddressbyPrefix(*prefix*, *status*, *network_id*, *callback=None*, *errback=None*)
Load an existing address by prefix, status and network into a high level Address object

Parameters

- **prefix** (*str*) – CIDR prefix of an existing Address
- **status** (*str*) – The status of address assignment (planned or assigned)
- **network_id** (*int*) – network_id associated with the address

loadDHCPOptions(*address_family*, *options*)
Create a high level DHCPOptions object

Parameters

- **address_family** (*str*) – Address family of the options. Can be either dhcpv4 or dhcpv6
- **options** (*dict*) – Dictionary containing the option set to apply for this address family. Note: only those specified will be applied. Allowed options can be found in *ns1.ipam.DHCPOptions.OPTIONS*

loadLeases(*scope_group_id=None*, *scope_id=None*, *limit=None*, *offset=None*, *callback=None*, *errback=None*)

loadMonitors(*callback=None*, *errback=None*, ***kwargs*)
Load all monitors

loadNetworkbyID(*id*, *callback=None*, *errback=None*)
Load an existing Network by ID into a high level Network object

Parameters **id** (*int*) – id of an existing Network

loadNetworkbyName(*name*, *callback=None*, *errback=None*)
Load an existing Network by name into a high level Network object

Parameters **name** (*str*) – Name of an existing Network

loadRecord(*domain*, *type*, *zone=None*, *callback=None*, *errback=None*, ***kwargs*)
Load an existing record into a high level Record object.

Parameters

- **domain** (*str*) – domain name of the record in the zone, for example 'myrecord'. You may leave off the zone, if it is specified in the zone parameter. This is recommended. You can pass a fully qualified domain and not pass the zone argument, but this will not work as expected if there are any dots in the domain, e.g. *foo.example.com* is OK, *foo.bar.example.com* will not work as expected.
- **type** (*str*) – record type, such as 'A', 'MX', 'AAAA', etc.
- **zone** (*str*) – zone name, like 'example.com'

Return type *ns1.records*

loadReservation(*scopegroup_id*, *address_id*, *reservation_id=None*, *callback=None*, *errback=None*)

loadScope(*scopegroup_id*, *address_id*, *callback=None*, *errback=None*)

loadScopeGroup(*id*, *callback=None*, *errback=None*)
Load an existing Scope Group into a high level Scope Group object

Parameters **id** (*int*) – id of an existing ScopeGroup

loadZone(*zone*, *callback=None*, *errback=None*)
Load an existing zone into a high level Zone object.

Parameters **zone** (*str*) – zone name, like 'example.com'

Return type *ns1.zones.Zone*

monitoring_jobtypes()
Return a new raw REST interface to monitoring jobtypes resources

Return type ns1.rest.monitoring.JobTypes

monitoring_regions()
Return a new raw REST interface to monitoring regions resources

Return type ns1.rest.monitoring.Regions

monitors()
Return a new raw REST interface to monitors resources

Return type ns1.rest.monitoring.Monitors

networks()
Return a new raw REST interface to network resources

Return type *ns1.rest.ipam.Networks*

notifylists()
Return a new raw REST interface to notify list resources

Return type ns1.rest.monitoring.NotifyLists

optiondefs()
Return a new raw REST interface to optiondefs resources

Return type *ns1.rest.ipam.Optiondefs*

plan()
Return a new raw REST interface to account plan

Return type ns1.rest.account.Plan

pools()
Return a new raw REST interface to Pools resources

Return type ns1.rest.pools.Pools

records()
Return a new raw REST interface to record resources

Return type *ns1.rest.records.Records*

reservations()
Return a new raw REST interface to reservation resources

Return type *ns1.rest.ipam.Reservations*

scope_groups()
Return a new raw REST interface to scope_group resources

Return type *ns1.rest.ipam.Scopegroups*

scopes()
Return a new raw REST interface to scope resources

Return type *ns1.rest.ipam.Scopes*

searchZone(*query*, *type='all'*, *expand=True*, *max=None*, *callback=None*, *errback=None*)
This method was updated since NS1 deprecated v1/search/zones Search a zone record or answers for a given search query (e.g., for geological data, etc)

Parameters

- **query** – query to search zone name or other type name
- **type** – String Filters search results by type. Enum: "zone", "record", "all", "answers"
- **expand** – Boolean Expands contents of search results.

- **max** – Integer Maximum number of search results to display

Returns list of zones searched

stats()
Return a new raw REST interface to stats resources

Return type *ns1.rest.stats.Stats*

team()
Return a new raw REST interface to team resources

Return type *ns1.rest.team.Team*

tsig()
Return a new raw REST interface to tsgi resources

Return type ns1.rest.tsig.Tsgi

user()
Return a new raw REST interface to user resources

Return type *ns1.rest.user.User*

views()
Return a new raw REST interface to View resources

Return type ns1.rest.views.Views

zones()
Return a new raw REST interface to zone resources

Return type *ns1.rest.zones.Zones*

5.1.2 ns1.config

This object is used to configure the SDK and REST client. It handles multiple API keys via a simple selection mechanism (keyID).

Sample:

```
{
   "default_key": "account2",
   "verbosity": 5,
   "keys": {
        "account1": {
            "key": "<<CLEARTEXT API KEY>>",
            "desc": "account number 1",
            "writeLock": true
        },
        "account2": {
            "key": "<<ANOTHER CLEARTEXT API KEY>>",
            "desc": "account number 2",
            "writeLock": false
        }
   },
   "cli": {
       "output_format": "text"
   }
}
```

class ns1.config.**Config**(*path=None*)

A simple object for accessing and manipulating config files. These contains options and credentials for accessing the NS1 REST API. Config files are simple JSON text files. To set or retrieve vales, use the object like a dict.

Parameters **path** (*str*) – optional path. if given, try to load the given config file

API_VERSION = 'v1'

DEFAULT_CONFIG_FILE = '~/.nsone'

ENDPOINT = 'api.nsone.net'

PORT = 443

createFromAPIKey(*apikey*, *maybeWriteDefault=False*)

Create a basic config from a single API key

Parameters

- **apikey** (*str*) – NS1 API Key, as created in the NS1 portal
- **maybeWriteDefault** (*bool*) – If True and DEFAULT_CONFIG_FILE doesn't exist write out the resulting config there.

get(*item*, *default=None*)

Retrieve a value from the config object.

Parameters

- **item** (*str*) – Key to lookup
- **default** – Default value to return if the requested item doesn't exist

Returns Requested value, or *default* if it didn't exist

getAPIKey(*keyID=None*)

Retrieve the NS1 API Key for the given keyID

Parameters **keyID** (*str*) – optional keyID to retrieve, or current if not passed

Returns API Key for the given keyID

getCurrentKeyID()

Retrieve the current keyID in use.

Returns current keyID in use

getEndpoint()

Retrieve the NS1 API Endpoint URL that will be used for requests.

Returns URL of the NS1 API that will be used for requests

getKeyConfig(*keyID=None*)

Get key configuration specified by *keyID*, or current keyID.

Parameters **keyID** (*str*) – optional keyID to retrieve, or current if not passed

Returns a dict of the request (or current) key config

getRateLimitingFunc()

choose how to handle rate limiting

isKeyWriteLocked(*keyID=None*)

Determine if a key config is write locked.

Parameters **keyID** (*str*) – optional keyID to retrieve, or current if not passed

Returns True if the given (or current) keyID is writeLocked

loadFromDict(*d*)
Load config data from the given dictionary

Parameters **d** (*dict*) – Python dictionary containing configuration items

loadFromFile(*path*)
Load JSON config file from disk at the given path

Parameters **path** (*str*) – path to config file

loadFromString(*body*)
Load config data (i.e. JSON text) from the given string

Parameters **body** (*str*) – config data in JSON format

useKeyID(*keyID*)
Use the given API key config specified by *keyID* during subsequent API calls

Parameters **keyID** (*str*) – an index into the 'keys' maintained in this config

write(*path=None*)

Write config data to disk. If this config object already has a path, it will write to it. If it doesn't, one must be passed during this call.

Parameters **path** (*str*) – path to config file

exception ns1.config.**ConfigException**
Bases: exceptions.Exception

5.1.3 ns1.zones

Object representing a single DNS zone.

Note: Answers to a record (the *answers* kwarg) should be passed as one of the following four structures, depending on how advanced the configuration for the answer needs to be:

1. A single string that is coerced to a single answer with no other fields e.g. meta. For example: *"1.1.1.1"*
2. A list of single strings that is coerced to several answers with no other fields e.g. meta. For example: *["1.1.1.1", "2.2.2.2"]*
3. A list of lists. In this case there will be as many answers as are in the outer list, and the answers themselves are used verbatim from the inner list (e.g. may have MX style *[10, '1.1.1.1]*), but no other fields e.g. meta. You must use this form for MX records, and if there is only one answer it still must be wrapped in an outer list.
4. A list of dicts. In this case it expects the full rest model and passes it along unchanged. You must use this form for any advanced record config like meta data or data feeds.

```
# Example of an advanced answer configuration (list of dicts)
record = yield zone.add_A('record',
                          [{'answer': ['1.1.1.1'],
                            'meta': {
                                'up': False
                                }
                            },
                           {'answer': ['9.9.9.9'],
                            'meta': {
```

(continues on next page)

(continued from previous page)

```
                                    'up': True
                                    }
                                }],
                            filters=[{'up': {}}])
```

class ns1.zones.**Zone**(*config*, *zone*)

Bases: object

High level object representing a Zone. In addition to the documented methods, there are magic methods allowing easy creation of records in this zone. Simply can 'add_TYPE' where TYPE is a valid DNS record type, such as add_A(). See examples for more information.

Create a new high level Zone object

Parameters

- **config** (ns1.config.Config) – config object
- **zone** (*str*) – zone name

cloneRecord(*existing_domain*, *new_domain*, *rtype*, *zone=None*, *callback=None*, *errback=None*)

Clone the given record to a new record such that their configs are identical.

Parameters

- **existing_domain** (*str*) – The existing record to clone
- **new_domain** (*str*) – The name of the new cloned record
- **rtype** (*str*) – DNS record type
- **zone** (*str*) – Optional zone name, if the new record should exist in a different zone than the original record.

Return type *ns1.records.Record*

Returns new Record

create(*zoneFile=None*, *callback=None*, *errback=None*, ***kwargs*)

Create a new zone. Pass a list of keywords and their values to configure. For the list of keywords available for zone configuration, see *ns1.rest.zones.Zones.INT_FIELDS* and *ns1.rest.zones.Zones.PASSTHRU_FIELDS* If zoneFile is passed, it should be a zone text file on the local disk that will be used to populate the created zone file.

createLinkToSelf(*new_zone*, *callback=None*, *errback=None*, ***kwargs*)

Create a new linked zone, linking to ourselves. All records in this zone will then be available as "linked records" in the new zone.

Parameters **new_zone** (*str*) – the new zone name to link to this one

Returns new Zone

delete(*callback=None*, *errback=None*)

Delete the zone and ALL records it contains.

linkRecord(*existing_domain*, *new_domain*, *rtype*, *callback=None*, *errback=None*, ***kwargs*)

Create a new linked record in this zone. These records use the configuration (answers, ttl, filters, etc) from an existing record in the NS1 platform.

Parameters

- **existing_domain** (*str*) – FQDN of the target record whose config should be used. Does not have to be in the same zone.

- **new_domain** (*str*) – Name of the new (linked) record. Zone name is appended automatically.
- **rtype** (*str*) – DNS record type, which must match the target record.

Return type *ns1.records.Record*

Returns new Record

load(*callback=None*, *errback=None*, *reload=False*)
Load zone data from the API.

loadRecord(*domain*, *rtype*, *callback=None*, *errback=None*)
Load a high level Record object from a domain within this Zone.

Parameters

- **domain** (*str*) – The name of the record to load
- **rtype** (*str*) – The DNS record type

Return type *ns1.records.Record*

Returns new Record

qps(*callback=None*, *errback=None*)
Return the current QPS for this zone

Return type dict

Returns QPS information

reload(*callback=None*, *errback=None*)
Reload zone data from the API.

update(*callback=None*, *errback=None*, ***kwargs*)
Update zone configuration. Pass a list of keywords and their values to update. For the list of keywords available for zone configuration, see *ns1.rest.zones.Zones.INT_FIELDS* and *ns1.rest.zones.Zones.PASSTHRU_FIELDS*

usage(*callback=None*, *errback=None*, ***kwargs*)
Return the current usage information for this zone

Return type dict

Returns usage information

exception ns1.zones.**ZoneException**
Bases: exceptions.Exception

5.1.4 ns1.records

Object representing a single DNS record in a zone of a specific type.

Note: Answers to a record (the *answers* kwarg) should be passed as one of the following four structures, depending on how advanced the configuration for the answer needs to be:

1. A single string that is coerced to a single answer with no other fields e.g. meta. For example: *"1.1.1.1"*
2. A list of single strings that is coerced to several answers with no other fields e.g. meta. For example: *["1.1.1.1", "2.2.2.2"]*

3. A list of lists. In this case there will be as many answers as are in the outer list, and the answers themselves are used verbatim from the inner list (e.g. may have MX style *[10, '1.1.1.1]*), but no other fields e.g. meta. You must use this form for MX records, and if there is only one answer it still must be wrapped in an outer list.
4. A list of dicts. In this case it expects the full rest model and passes it along unchanged. You must use this form for any advanced record config like meta data or data feeds.

```
# Example of an advanced answer configuration (list of dicts)
record = yield zone.add_A('record',
                          [{'answer': ['1.1.1.1'],
                            'meta': {
                                'up': False
                                }
                            },
                           {'answer': ['9.9.9.9'],
                            'meta': {
                                'up': True
                                }
                            }],
                          filters=[{'up': {}}])
```

class ns1.records.**Record**(*parentZone*, *domain*, *type*)

Bases: object

High level object representing a Record

Create a new high level Record

Parameters

- **parentZone** (ns1.zones.Zone) – the high level Zone parent object
- **domain** (*str*) – full domain name this record represents. if the domain does not end with the zone name, it is appended.
- **type** (*str*) – The DNS record type (A, MX, etc)

addAnswers(*answers*, *callback=None*, *errback=None*, ***kwargs*)

Add answers to the record.

Parameters **answers** – answers structure. See the class note on answer format.

create(*callback=None*, *errback=None*, ***kwargs*)

Create new record. Pass a list of keywords and their values to config. For the list of keywords available for zone configuration, see *ns1.rest.records.Records.INT_FIELDS*, *ns1.rest.records.Records.PASSTHRU_FIELDS*, *ns1.rest.records.Records.BOOL_FIELDS*

delete(*callback=None*, *errback=None*)

Delete the record from the zone, including all advanced configuration, meta data, etc.

load(*callback=None*, *errback=None*, *reload=False*)

Load record data from the API.

qps(*callback=None*, *errback=None*)

Return the current QPS for this record

Return type dict

Returns QPS information

reload(*callback=None*, *errback=None*)

Reload record data from the API.

update(*callback=None*, *errback=None*, ***kwargs*)
Update record configuration. Pass list of keywords and their values to update. For the list of keywords available for zone configuration, see *ns1.rest.records.Records.INT_FIELDS*, *ns1.rest.records.Records.PASSTHRU_FIELDS*, *ns1.rest.records.Records.BOOL_FIELDS*

usage(*callback=None*, *errback=None*, ***kwargs*)
Return the current usage information for this record

Return type dict

Returns usage information

exception ns1.records.**RecordException**
Bases: exceptions.Exception

5.1.5 ns1.ipam

A collection of classes that produce IPAM Objects.

class ns1.ipam.**Address**(*config*, *prefix=None*, *status=None*, *network=None*, *scope_group=None*, *id=None*, *tags=None*)
Bases: object

Create a new high level Address object

Parameters

- **config** (ns1.config.Config) – config object
- **prefix** (*str*) – cidr prefix
- **status** (*str*) – planned, assigned
- **network** (Network) – Network Object the address will be part of
- **scope_group** (Scopegroup) – Scopegroup Object that will be associated with the address
- **tags** (*dict*) – tags of the address

create(*callback=None*, *errback=None*, *parent=True*, ***kwargs*)
Create a new Address. Pass a list of keywords and their values to configure. For the list of keywords available for address configuration,see *ns1.rest.ipam.Addresses.INT_FIELDS* and *ns1.rest.ipam.Addresses.PASSTHRU_FIELDS*

delete(*callback=None*, *errback=None*)
Delete the address and all child addresses

load(*callback=None*, *errback=None*, *reload=False*)
Load address data from the API.

reload(*callback=None*, *errback=None*)
Reload address data from the API.

reserve(*scopegroup_id*, *mac*, *options=None*, *callback=None*, *errback=None*)
Add scope group reservation. Pass a single Address object and a MAC address as a string

update(*callback=None*, *errback=None*, *parent=True*, ***kwargs*)
Update address configuration. Pass a list of keywords and their values to update. For the list of keywords available for address configuration, see *ns1.rest.ipam.Addresses.INT_FIELDS* and *ns1.rest.ipam.Addresses.PASSTHRU_FIELDS*

exception ns1.ipam.**AddressException**
Bases: exceptions.Exception

class ns1.ipam.**DHCPOptionValue**(*key*, *value*, *always_send=None*)
Create the DHCPOptionValue class that can be used as value with *ns1.ipam.DHCPOptions*

:param key str option name :param value any option value :param always_send bool indicates whether this option be sent back in lease or not

generate_option(*address_family*)
Generates dhcp option value with a proper format

:param address_family str one of dhcpv4 or dhcpv6 family name

class ns1.ipam.**DHCPOptions**(*address_family*, *options*, *server_options=None*)
Create the DHCP options class that can be used by the IPAM API

Parameters

- **address_family** (*str*) – This is either dhcpv4 or dhcpv6
- **options** (*list*) – This is a list of ns1.ipam.DHCPOptionsValue objects representing the DHCP options

AF = ['dhcpv4', 'dhcpv6']

OPTIONS = {'dhcpv4': ['bootfile-name', 'domain-name', 'domain-name-servers', 'host-na

update(*address_family*, *options*, *server_options=None*)

exception ns1.ipam.**DHCPOptionsException**
Bases: exceptions.Exception

class ns1.ipam.**Lease**(*config*)
Bases: object

Create a new high level Lease object

Parameters **config** (ns1.config.Config) – config object

load(*scope_group_id=None*, *scope_id=None*, *limit=None*, *offset=None*, *callback=None*, *errback=None*, *reload=False*)
Load Lease data from the API.

reload(*callback=None*, *errback=None*)
Reload Lease data from the API.

exception ns1.ipam.**LeaseException**
Bases: exceptions.Exception

class ns1.ipam.**Network**(*config*, *name=None*, *id=None*, *tags=None*)
Bases: object

Create a new high level Network object

Parameters

- **config** (ns1.config.Config) – config object
- **name** (*str*) – network name
- **id** (*int*) – id of an existing Network
- **tags** (*dict*) – tags of the network

create(*callback=None*, *errback=None*, ***kwargs*)
Create a new Network. Pass a list of keywords and their values to configure. For the list of keywords available for network configuration, see *ns1.rest.ipam.Networks.INT_FIELDS* and *ns1.rest.ipam.Networks.PASSTHRU_FIELDS*

delete(*callback=None*, *errback=None*)
Delete the Network and all associated addresses

load(*callback=None*, *errback=None*, *reload=False*)
Load network data from the API.

new_address(*prefix*, *status*, *callback=None*, *errback=None*, ***kwargs*)
Create a new address space in this Network

Parameters

- **prefix** (*str*) – The CIDR prefix of the address to add
- **status** (*str*) – planned, assigned

Returns The newly created Address object

reload(*callback=None*, *errback=None*)
Reload network data from the API.

update(*callback=None*, *errback=None*, ***kwargs*)
Update Network configuration. Pass a list of keywords and their values to update. For the list of keywords available for zone configuration, see *ns1.rest.ipam.Networks.INT_FIELDS* and *ns1.rest.ipam.Networks.PASSTHRU_FIELDS*

exception ns1.ipam.**NetworkException**
Bases: exceptions.Exception

class ns1.ipam.**Optiondef**(*config*, *space*, *key*)
Bases: object

Create a new high level Optiondef object

Parameters

- **config** (ns1.config.Config) – config object
- **space** (*str*) – dhcpv4 or dhcpv6
- **key** (*str*) – option key

create(*callback=None*, *errback=None*, ***kwargs*)
Create a new Optiondef. Pass a list of keywords and their values to configure. For the list of keywords available for address configuration, see ns1.rest.ipam.Optiondef.INT_FIELDS and ns1.rest.ipam.Optiondef.PASSTHRU_FIELDS

delete(*callback=None*, *errback=None*)
Delete the Optiondef

load(*callback=None*, *errback=None*, *reload=False*)
Load Optiondef data from the API.

reload(*callback=None*, *errback=None*)
Reload OptionDef data from the API.

exception ns1.ipam.**OptiondefException**
Bases: exceptions.Exception

class ns1.ipam.**Reservation**(*config*, *scopegroup_id*, *address_id*, *reservation_id=None*, *options=None*, *mac=None*, *tags=None*)

Bases: object

Create a new high level Reservation object

Parameters

- **config** (ns1.config.Config) – config object
- **scopegroup_id** (*int*) – id of the scope group
- **address_id** (*int*) – id of the address the reservation is associated with
- **reservation_id** (*int*) – id of the reservation
- **options** (*list*) – dhcp options of the reservation
- **mac** (*str*) – mac address of the reservation
- **tags** (*dict*) – tags of the reservation

create(*callback=None*, *errback=None*, ***kwargs*)
Create a new Reservation. Pass a list of keywords and their values to configure. For the list of keywords available for address configuration, see *ns1.rest.ipam.Reservations.INT_FIELDS* and *ns1.rest.ipam.Reservations.PASSTHRU_FIELDS*

delete(*callback=None*, *errback=None*)
Delete the Reservation

load(*callback=None*, *errback=None*, *reload=False*)
Load Reservation data from the API.

reload(*callback=None*, *errback=None*)
Reload Reservation data from the API.

update(*options*, *callback=None*, *errback=None*, *parent=True*, ***kwargs*)
Update reservation configuration. Pass a list of keywords and their values to update. For the list of keywords available for address configuration, see *ns1.rest.ipam.Reservations.INT_FIELDS* and *ns1.rest.ipam.Reservations.PASSTHRU_FIELDS*

exception ns1.ipam.**ReservationException**

Bases: exceptions.Exception

class ns1.ipam.**Scope**(*config*, *scopegroup_id*, *address_id*, *scope_id=None*, *options=None*, *tags=None*)

Bases: object

Create a new high level Scope object

Parameters

- **config** (ns1.config.Config) – config object
- **scopegroup_id** (*int*) – id of the scope group
- **address_id** (*int*) – id of the address the scope is associated with
- **scope_id** (*int*) – id of the scope
- **options** (DHCPOptions) – DHCPOptions object that contains the settings for the scope
- **tags** (*dict*) – tags of the scope

create(*callback=None*, *errback=None*, ***kwargs*)
Create a new Scope. Pass a list of keywords and their values to configure. For the list of keywords available for address configuration, see `ns1.rest.ipam.Scope.INT_FIELDS` and *ns1.rest.ipam.Reservations.PASSTHRU_FIELDS*

delete(*callback=None*, *errback=None*)
Delete the Scope

load(*callback=None*, *errback=None*, *reload=False*)
Load Reservation data from the API.

reload(*callback=None*, *errback=None*)
Reload Scope data from the API.

update(*address_id*, *options*, *callback=None*, *errback=None*, ***kwargs*)
Update Scope configuration. Pass a list of keywords and their values to update. For the list of keywords available for address configuration, see *ns1.rest.ipam.Scopes.INT_FIELDS* and *ns1.rest.ipam.Scopes.PASSTHRU_FIELDS*

exception ns1.ipam.**ScopeException**
Bases: exceptions.Exception

class ns1.ipam.**Scopegroup**(*config*, *name=None*, *service_def_id=None*, *id=None*, *tags=None*)
Bases: object

Create a new high level Scopegroup object

Parameters

- **config** (ns1.config.Config) – config object
- **name** (*str*) – Name of the scope group
- **service_group_id** (*int*) – id of the service group the scope group is associated with
- **id** (*int*) – id of the scope group
- **tags** (*dict*) – tags of the scopegroup

create(*dhcp4*, *dhcp6*, *callback=None*, *errback=None*, ***kwargs*)

Parameters

- **dhcp4** (DHCPOptions) – DHCPOptions object that contains the settings for dhcp4
- **dhcp6** (DHCPOptions) – DHCPOptions object that contains the settings for dhcp6

Create a new Scope Group. Pass a list of keywords and their values to configure. For the list of keywords available for address configuration, see *ns1.rest.ipam.Scopegroups.INT_FIELDS* and *ns1.rest.ipam.Scopegroups.PASSTHRU_FIELDS*. For the list of settings see `ns1.ipan.Scopegroup.SETTINGS`. Note that if *enabled* is True, then *valid_lifetime_secs* must be set to a value greater than 0.

create_scope(*address_id*, *callback=None*, *errback=None*)
Add scope group scope. Pass a single Address ID

delete(*callback=None*, *errback=None*)
Delete the Scopegroup and all child addresses

load(*callback=None*, *errback=None*, *reload=False*)
Load Scopegroup data from the API.

reload(*callback=None*, *errback=None*)
Reload Scopegroup data from the API.

reservations

reserve(*address_id*, *mac*, *options=None*, *callback=None*, *errback=None*)

Parameters

- **address_id** (*int*) – id of the Address to reserve
- **options** (DHCPOptions) – DHCPOptions object that contains the settings for the address
- **str** (*mac*) – MAC address of the reservation

Add scope group reservation. Pass a single Address ID and a MAC address as a string

scopes

update(*callback=None*, *errback=None*, ***kwargs*)

Update scope group configuration. Pass a list of keywords and their values to update. For the list of keywords available for address configuration, see *ns1.rest.ipam.Scopegroups.INT_FIELDS* and *ns1.rest.ipam.Scopegroups.PASSTHRU_FIELDS*

exception ns1.ipam.**ScopegroupException**

Bases: exceptions.Exception

5.1.6 ns1.rest

A thin layer over the NS1 REST API

exception ns1.rest.errors.**AuthException**(*message*, *response=None*, *body=None*)

Bases: *ns1.rest.errors.ResourceException*

exception ns1.rest.errors.**RateLimitException**(*message*, *response=None*, *body=None*, *by=None*, *limit=None*, *remaining=None*, *period=None*)

Bases: *ns1.rest.errors.ResourceException*

exception ns1.rest.errors.**ResourceException**(*message*, *response=None*, *body=None*)

Bases: exceptions.Exception

class ns1.rest.resource.**BaseResource**(*config*)

Parameters **config** (ns1.config.Config) – config object used to build requests

BOOL_FIELDS = []

DEFAULT_TRANSPORT = 'requests'

INT_FIELDS = []

PASSTHRU_FIELDS = []

class ns1.rest.data.**Feed**(*config*)

Bases: *ns1.rest.resource.BaseResource*

Parameters **config** (ns1.config.Config) – config object used to build requests

PASSTHRU_FIELDS = ['name', 'config']

ROOT = 'data/feeds'

create(*sourceid*, *name*, *config*, *callback=None*, *errback=None*, ***kwargs*)

delete(*sourceid*, *feedid*, *callback=None*, *errback=None*)

list(*sourceid*, *callback=None*, *errback=None*)

retrieve(*sourceid*, *feedid*, *callback=None*, *errback=None*)

update(*sourceid*, *feedid*, *callback=None*, *errback=None*, ***kwargs*)

class ns1.rest.data.**Source**(*config*)
Bases: *ns1.rest.resource.BaseResource*

Parameters **config** (ns1.config.Config) – config object used to build requests

PASSTHRU_FIELDS = ['name', 'config']

ROOT = 'data/sources'

create(*name*, *sourcetype*, *callback=None*, *errback=None*, ***kwargs*)
The only supported kwarg is *config*.

delete(*sourceid*, *callback=None*, *errback=None*)

list(*callback=None*, *errback=None*)

publish(*sourceid*, *data*, *callback=None*, *errback=None*)

retrieve(*sourceid*, *callback=None*, *errback=None*)

update(*sourceid*, *sourcetype*, *callback=None*, *errback=None*, ***kwargs*)
Note that *sourcetype* is required, but cannot be changed by this method.

Supported kwargs are: *name*, *config*.

class ns1.rest.stats.**Stats**(*config*)
Bases: *ns1.rest.resource.BaseResource*

Parameters **config** (ns1.config.Config) – config object used to build requests

ROOT = 'stats'

qps(*zone=None*, *domain=None*, *type=None*, *callback=None*, *errback=None*)

usage(*zone=None*, *domain=None*, *type=None*, *callback=None*, *errback=None*, ***kwargs*)

ns1.rest.stats.**stats_usage_pagination**(*curr_json*, *next_json*)

class ns1.rest.records.**Records**(*config*)
Bases: *ns1.rest.resource.BaseResource*

Parameters **config** (ns1.config.Config) – config object used to build requests

BOOL_FIELDS = ['use_client_subnet', 'use_csubnet', 'override_ttl']

INT_FIELDS = ['ttl']

PASSTHRU_FIELDS = ['networks', 'meta', 'regions', 'link']

ROOT = 'zones'

create(*zone*, *domain*, *type*, *callback=None*, *errback=None*, ***kwargs*)

create_raw(*zone*, *domain*, *type*, *body*, *callback=None*, *errback=None*, ***kwargs*)

delete(*zone*, *domain*, *type*, *callback=None*, *errback=None*)

retrieve(*zone*, *domain*, *type*, *callback=None*, *errback=None*)

update(*zone*, *domain*, *type*, *callback=None*, *errback=None*, ***kwargs*)

class ns1.rest.zones.**Zones**(*config*)
Bases: *ns1.rest.resource.BaseResource*

Parameters **config** (ns1.config.Config) – config object used to build requests

BOOL_FIELDS = ['dnssec']

INT_FIELDS = ['ttl', 'retry', 'refresh', 'expiry', 'nx_ttl']

PASSTHRU_FIELDS = ['primary', 'secondary', 'hostmaster', 'meta', 'networks', 'link', '

ROOT = 'zones'

SEARCH_ROOT = 'search'

activate_version(*zone*, *version_id*, *callback=None*, *errback=None*)

create(*zone*, *callback=None*, *errback=None*, ***kwargs*)

create_version(*zone*, *force=False*, *callback=None*, *errback=None*)

delete(*zone*, *callback=None*, *errback=None*)

delete_version(*zone*, *version_id*, *callback=None*, *errback=None*)

import_file(*zone*, *zoneFile*, *callback=None*, *errback=None*, ***kwargs*)

list(*callback=None*, *errback=None*)

list_versions(*zone*, *callback=None*, *errback=None*)

retrieve(*zone*, *callback=None*, *errback=None*)

search(*query*, *type='all'*, *expand=True*, *max=None*, *callback=None*, *errback=None*)

update(*zone*, *callback=None*, *errback=None*, ***kwargs*)

ns1.rest.zones.**zone_list_pagination**(*curr_json*, *next_json*)

ns1.rest.zones.**zone_retrieve_pagination**(*curr_json*, *next_json*)

class ns1.rest.ipam.**Addresses**(*config*)

Bases: *ns1.rest.resource.BaseResource*

Parameters **config** (ns1.config.Config) – config object used to build requests

BOOL_FIELDS = ['parent']

INT_FIELDS = ['network_id', 'address_id', 'root_address_id', ' merged_address_id', 'sc

PASSTHRU_FIELDS = ['prefix', 'status', 'desc', 'tags', 'reserve']

ROOT = 'ipam/address'

create(*callback=None*, *errback=None*, *parent=True*, ***kwargs*)

delete(*address_id*, *callback=None*, *errback=None*)

list(*callback=None*, *errback=None*)

report(*address_id*, *callback=None*, *errback=None*)

retrieve(*address_id*, *callback=None*, *errback=None*)

retrieve_children(*address_id*, *callback=None*, *errback=None*)

retrieve_dhcp_option(*address_id*, *callback=None*, *errback=None*)

retrieve_parent(*address_id*, *callback=None*, *errback=None*)

search(*network_id*, *prefix*, *callback=None*, *errback=None*)

update(*address_id*, *callback=None*, *errback=None*, *parent=True*, ***kwargs*)

class ns1.rest.ipam.**Leases**(*config*)

Bases: *ns1.rest.resource.BaseResource*

Parameters **config** (ns1.config.Config) – config object used to build requests

BOOL_FIELDS = []

INT_FIELDS = ['scope_group_id', 'scope_id', 'limit', 'offset']

PASSTHRU_FIELDS = []

ROOT = 'dhcp/lease'

list(*scope_group_id=None*, *scope_id=None*, *limit=None*, *offset=None*, *callback=None*, *errback=None*)

class ns1.rest.ipam.**Networks**(*config*)

Bases: *ns1.rest.resource.BaseResource*

Parameters **config** (ns1.config.Config) – config object used to build requests

BOOL_FIELDS = []

INT_FIELDS = ['network_id']

PASSTHRU_FIELDS = ['rt', 'name', 'desc', 'tags']

ROOT = 'ipam/network'

create(*callback=None*, *errback=None*, ***kwargs*)

delete(*network_id*, *callback=None*, *errback=None*)

list(*callback=None*, *errback=None*, *expand=True*)

report(*network_id*, *callback=None*, *errback=None*)

retrieve(*network_id*, *callback=None*, *errback=None*, *expand=True*)

update(*network_id*, *callback=None*, *errback=None*, ***kwargs*)

class ns1.rest.ipam.**Optiondefs**(*config*)

Bases: *ns1.rest.resource.BaseResource*

Parameters **config** (ns1.config.Config) – config object used to build requests

BOOL_FIELDS = ['standard']

INT_FIELDS = ['code']

PASSTHRU_FIELDS = ['space', 'key', 'friendly_name', 'description', 'schema']

ROOT = 'dhcp/optiondef'

create(*space*, *key*, *callback=None*, *errback=None*, ***kwargs*)

delete(*space*, *key*, *callback=None*, *errback=None*)

list(*callback=None*, *errback=None*)

retrieve(*space*, *key*, *callback=None*, *errback=None*)

class ns1.rest.ipam.**Reservations**(*config*)

Bases: *ns1.rest.resource.BaseResource*

Parameters **config** (ns1.config.Config) – config object used to build requests

BOOL_FIELDS = ['dhcpv6']

INT_FIELDS = ['scope_group_id', 'address_id']

PASSTHRU_FIELDS = ['mac', 'options', 'tags']

ROOT = 'dhcp/reservation'

create(*scopegroup_id*, *address_id*, *options*, *callback=None*, *errback=None*, ***kwargs*)

delete(*reservation_id*, *callback=None*, *errback=None*)

list(*scopegroup_id*, *callback=None*, *errback=None*)

retrieve(*reservation_id*, *callback=None*, *errback=None*)

classmethod select_from_list(*result*, *address_id*)

update(*reservation_id*, *options*, *callback=None*, *errback=None*, ***kwargs*)

class ns1.rest.ipam.**Scopegroups**(*config*)
Bases: *ns1.rest.resource.BaseResource*

Parameters **config** (ns1.config.Config) – config object used to build requests

BOOL_FIELDS = ['enabled', 'echo_client_id']

INT_FIELDS = ['id', 'dhcp_service_id', 'valid_lifetime_secs']

PASSTHRU_FIELDS = ['dhcpv4', 'dhcpv6', 'name', 'tags']

ROOT = 'dhcp/scopegroup'

create(*callback=None*, *errback=None*, ***kwargs*)

delete(*scopegroup_id*, *callback=None*, *errback=None*)

list(*callback=None*, *errback=None*)

retrieve(*scope_group_id*, *callback=None*, *errback=None*)

update(*scope_group_id*, *callback=None*, *errback=None*, ***kwargs*)

class ns1.rest.ipam.**Scopes**(*config*)
Bases: *ns1.rest.resource.BaseResource*

Parameters **config** (ns1.config.Config) – config object used to build requests

INT_FIELDS = ['scope_group_id', 'address_id', 'valid_lifetime_secs']

PASSTHRU_FIELDS = ['options', 'tags']

ROOT = 'dhcp/scope'

create(*scopegroup_id*, *address_id*, *options*, *callback=None*, *errback=None*, ***kwargs*)

delete(*scope_id*, *callback=None*, *errback=None*)

list(*scopegroup_id*, *callback=None*, *errback=None*)

retrieve(*scope_id*, *callback=None*, *errback=None*)

classmethod select_from_list(*result*, *scope_id*)

update(*scope_id*, *address_id*, *options*, *scopegroup_id=None*, *callback=None*, *errback=None*, ***kwargs*)

class ns1.rest.team.**Team**(*config*)
Bases: *ns1.rest.resource.BaseResource*

Parameters **config** (ns1.config.Config) – config object used to build requests

PASSTHRU_FIELDS = ['name', 'ip_whitelist', 'permissions']

ROOT = 'account/teams'

create(*name*, *callback=None*, *errback=None*, ***kwargs*)

delete(*team_id*, *callback=None*, *errback=None*)

list(*callback=None*, *errback=None*)

retrieve(*team_id*, *callback=None*, *errback=None*)

update(*team_id*, *callback=None*, *errback=None*, ***kwargs*)

class ns1.rest.user.**User**(*config*)

Bases: *ns1.rest.resource.BaseResource*

Parameters **config** (ns1.config.Config) – config object used to build requests

PASSTHRU_FIELDS = ['name', 'username', 'email', 'teams', 'notify', 'ip_whitelist', 'ip

ROOT = 'account/users'

create(*name*, *username*, *email*, *callback=None*, *errback=None*, ***kwargs*)

delete(*username*, *callback=None*, *errback=None*)

list(*callback=None*, *errback=None*)

retrieve(*username*, *callback=None*, *errback=None*)

update(*username*, *callback=None*, *errback=None*, ***kwargs*)

class ns1.rest.apikey.**APIKey**(*config*)

Bases: *ns1.rest.resource.BaseResource*

Parameters **config** (ns1.config.Config) – config object used to build requests

PASSTHRU_FIELDS = ['name', 'teams', 'ip_whitelist', 'ip_whitelist_strict', 'permission

ROOT = 'account/apikeys'

create(*name*, *callback=None*, *errback=None*, ***kwargs*)

delete(*apikey_id*, *callback=None*, *errback=None*)

list(*callback=None*, *errback=None*)

retrieve(*apikey_id*, *callback=None*, *errback=None*)

update(*apikey_id*, *callback=None*, *errback=None*, ***kwargs*)

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

n

Index

E

F

G

I

L

M

N

O

P

Q

R

S

T

U

V

W

Z