stingray Documentation

Release 0.1.0

Mark Troyer

Aug 15, 2018

Contents:

`python-stingray` is a python module for using the REST API provided by the Pulse Secure Virtual Traffic Manager load balancer, previously known as Stingray, Zeus, and Steelapp.

CHAPTER 1

Installing and Using the `stingray` Module

1.1 Installation

```
pip install python-stingray
```

1.2 Usage Examples

1.2.1 Connecting to a Stingray device

Creating a `Client()` object:

```
In [1]: import stingray.apiclient as sapi

In [2]: client = sapi.Client(host=stingray.example.com, port=9070, user=admin,␣
↪password=admin.password, api_version=5.2, ssl_verify=False)

In [3]: client.get_supported_versions()

Out[3]: [u'4.0', u'5.0', u'5.1', u'5.2']
```

All of the arguments for creating a client object can be set as environment variables so they don't have to be passed on a command line or included in code. Environment variables are:

- STINGRAY_HOST
- STINGRAY_PORT
- STINGRAY_USER
- STINGRAY_PASSWORD
- STINGRAY_API_VERSION
- STINGRAY_SSL_VERIFY

If not given, `port` defaults to `9070`, and `ssl_verify` defaults to `True`. If no `api_version` is given the client will query the device for supported versions and will choose the latest version available.

1.2.2 Device Statistics

Note: Status is not supported in API version 1.0

Get a `StatusAPI()` object from the client:

```
In [1]: status = client.get_status()
```

Statistics for a load balancer pool:

```
In [2]: status.statistic('pools', 'my_pool')

Out [2]:
{u'algorithm': u'roundrobin',
 u'bw_limit_bytes_drop': 0,
 u'bw_limit_pkts_drop': 0,
 u'bytes_in': 0,
 u'bytes_out': 0,
 u'conns_queued': 0,
 u'disabled': 0,
 u'draining': 0,
 u'max_queue_time': 0,
 u'mean_queue_time': 0,
 u'min_queue_time': 0,
 u'nodes': 1,
 u'persistence': u'none',
 u'queue_timeouts': 0,
 u'session_migrated': 0,
 u'state': u'active',
 u'total_conn': 0}
```

1.2.3 Pool Configurations

Get a `Pools` object:

```
In [1]: from stingray.config.pools import Pools

In [2]: pools = Pools.from_client(client)
```

List current pools:

```
In [3]: pools.pools

Out[3]:
{u'Pool1': u'/api/tm/5.2/config/active/pools/Pool1',
 u'Pool2': u'/api/tm/5.2/config/active/pools/Pool2',
 u'Pool3': u'/api/tm/5.2/config/active/pools/Pool3'}
```

Add a new pool:

```
In [4]: new_pool = pools.add('new_pool', nodes=['node1', 'node2'])
```

Configure a pool:

```
In [5]: pool = pools.get('Pool1')

In [6]: pool.nodes()

Out [6]:
{u'Node1': {u'node': u'Node1', u'state': u'active'},
 u'Node2': {u'node': u'Node2', u'state': u'active'}}

In [7]: pool.drain_node('Node2')

Out [7]:
{u'Node1': {
   u'state': u'active',
   u'health': u'alive',
   u'connections': 9,
   u'requests': 0},
 u'Node2': {
   u'state': u'draining',
   u'health': u'alive',
   u'connections': 0,
   u'requests': 0}}
```

Update arbitrary pool properties:

```
In [8]: pool.properties['connection']

Out [9]:
{u'max_connect_time': 4,
 u'max_connections_per_node': 0,
 u'max_queue_size': 0,
 u'max_reply_time': 30,
 u'queue_timeout': 10
}

In [10]: pool.properties['connection']['queue_timeout'] = 30

In [11]: pool.update()

In [12]: pool.properties['connection']

Out [12]:
{u'max_connect_time': 4,
 u'max_connections_per_node': 0,
 u'max_queue_size': 0,
 u'max_reply_time': 30,
 u'queue_timeout': 30
}
```

CHAPTER 2

stingray.apiclient

2.1 Client

class stingray.apiclient.**Client**(*host=None*, *port=None*, *user=None*, *password=None*, *api_version=None*, *ssl_verify=None*)

Bases: object

Client class to access the Stingray REST API.

get_config_endpoints()

Get all configuration endpoints.

Returns Endpoint names and paths

Return type (dict)

get_status()

Get a status object for the REST API status/local_tm/ endpoint.

Returns (StatusAPI)

get_supported_versions()

Query the REST API for the version(s) it supports.

Returns Supported versions

Return type (list)

update()

Convenience method to update the properties for an endpoint on the load balancer

2.2 StatusAPI

class stingray.apiclient.**StatusAPI**(*host=None*, *port=None*, *user=None*, *password=None*, *api_version=None*, *ssl_verify=None*)

Bases: *stingray.apiclient.Client*

Class for interacting with the `status/local_tm/` endpoints. Not supported in version 1.0 of the REST API

backup(*backup_name*)
Get the properties of an individual backup

Parameters **backup_name** (*str*) – The name of the backup to get

Returns Parameters for the backup

Return type (dict)

backups()
List current backups

Returns List of dicts with backup name and path

Return type (list)

classmethod from_client(*client*)

information()
Get version and uuid for the load balancer.

Returns tm_version and uuid

Return type (dict)

state()
Get state information for load balancer components.

Returns

List of dicts with state for error level, errors, failed nodes, license, pools, tip errors, and virtual servers

Return type (list)

statistic(*type*, *stat=None*, *stat_target=None*)
Get either the list of statistics for an endpoint, or the statistic data. Some endpoints have nested children, some have multiple nested children.

Parameters

- **type** (*str*) – The statistic type, or name, from the list of available statistics, e.g. cache, pools, etc.
- **stat** (*str*) – For single level nested stats, this is the name of the stat to get information on. For multiple levels of nesting this is the next level in the path.
- **stat_target** (*str*) – For multiple level nested stats this is the name of the stat to get information for.

Returns Available statistics for the type and their paths (dict): Data for the requested statistic

Return type (dict)

statistics()
Get the list of statistics for load balancer components

Returns Statistic type and path

Return type (dict)

CHAPTER 3

stingray.config

Modules for interacting with the Stingray configuration endpoints of the REST API. Contains classes for working with Pools, Traffic IP Groups, and Virtual Servers.

3.1 pools

3.1.1 Pools

class stingray.config.pools.**Pools**(*host=None*, *port=None*, *user=None*, *password=None*, *api_version=None*, *ssl_verify=None*)

Bases: *stingray.apiclient.Client*

Class for interacting with Pools via the REST API

add(*pool*, *nodes=None*, ***pool_props*)

Add a new load balancer pool

Parameters

- **pool** (*str*) – The name of the pool to add
- **nodes** (*list*) – List of nodes for the pool
- **pool_props** (*dict*) – Additional arguments to set properties of the pool at time of creation. Must be a dict in the form of:

```
{'section': {'key': 'value'}}
```

Returns The new pool

Return type (*Pool*)

delete(*pool*)

Delete a load balancer pool

Parameters **pool** (*str*) – The name of the pool to delete

Returns Response from the _api_delete method

Return type (dict)

classmethod from_client(*client*)

get(*pool*)
Get a Pool object for the request pool.

Parameters **pool** (*str*) – The name of the pool to get

Returns The requested pool

Return type (*Pool*)

3.1.2 Pool

class stingray.config.pools.**Pool**(*pool_name*, *pool_path*, *pool_properties=None*, *host=None*, *port=None*, *user=None*, *password=None*, *api_version=None*, *ssl_verify=None*)
Bases: *stingray.apiclient.Client*

Class for interacting with individual pools via the REST API

add_node(*node*, *state='active'*, *priority=1*, *weight=1*)
Add a new node to the pool

Parameters

- **node** (*str*) – The node to add. Must be in accepted pool node config format: <ip or dns name>:<port>
- **state** (*str*) – active, draining, or disabled. Default is active because it should be pretty rare to add a node in any other state.
- **priority** (*int*) – Load balancer priority for the node
- **weight** (*int*) – Load balancer weight for the node

Returns Pool nodes status

Return type (dict)

delete_node(*node*)
Delete a node from the pool

Parameters **node** (*str*) – The node to delete

Returns Pool nodes status

Return type (dict)

disable_node(*node*)
Disable a node in the pool

Parameters **node** (*str*) – The node to disable

Returns Pool nodes status

Return type (dict)

drain_node(*node*)
Set a node in the pool to draining status

Parameters **node** (*str*) – The node to drain

Returns Pool nodes status

Return type (dict)

enable_node(*node*)
Reenable a node in the pool

Parameters **node** (*str*) – The node to enable

Returns Pool nodes status

Return type (dict)

classmethod from_client(*client*, *pool_name*, *pool_path=None*, *pool_properties=None*)

nodes_status()
Get status info for the nodes in the pool. Some info is found in the node properties, some in the node statistics.

Returns

Nodes and their status, e.g.:

```
{
    u'10.0.0.1': {
        u'connections': 0,
        u'health': u'alive',
        u'requests': 0,
        u'state': u'active'
    }
}
```

Return type (dict)

statistics()
Get statistics for the pool

Returns Pool statistics

Return type (dict)

3.2 traffic_ip_groups

3.2.1 TrafficIPGroups

class stingray.config.traffic_ip_groups.**TrafficIPGroups**(*host=None*, *port=None*, *user=None*, *password=None*, *api_version=None*, *ssl_verify=None*)

Bases: *stingray.apiclient.Client*

Class for interacting with Traffic IP Groups via the REST API

add(*group*, *ipaddresses=None*, *machines=None*, *mode='singlehosted'*, ***group_props*)
Add a new traffic ip group.

Parameters

- **group** (*str*) – The traffic ip group name
- **ipaddresses** (*list*) – IP addresses to assign to the group

- **machines** (*list*) – Load balancers that can host the group's IP addresses. Default is the current load balancer (or load balancers if clustered).
- **mode** (*str*) – Method used to distribute traffic across the cluster. Default is `singlehosted`
- **group_props** (*dict*) – Additional arguments to set the properties of the traffic ip group at time of creation. Must be a dict in the form of:

```
{'section': {'key': 'value'}}
```

Returns The new traffic ip group

Return type (*TrafficIPGroup*)

delete(*group*)
Delete a traffic ip group

Parameters **group** (*str*) – The name of the traffic ip group to delete

Returns Response from the _api_delete method

Return type (dict)

classmethod from_client(*client*)

get(*group*)
Get a TrafficIPGroup object for the requested traffic ip group

Parameters **group** (*str*) – The name of the traffic ip group to get

Returns The requested traffic ip group

Return type (*TrafficIPGroup*)

3.2.2 TrafficIPGroup

class stingray.config.traffic_ip_groups.**TrafficIPGroup**(*group_name*, *group_path=None*, *group_properties=None*, *host=None*, *port=None*, *user=None*, *password=None*, *api_version=None*, *ssl_verify=None*)

Bases: *stingray.apiclient.Client*

Class for interacting with individual traffic ip groups via the REST API

classmethod from_client(*client*, *group_name*, *group_path=None*, *group_properties=None*)

statistics()
Get statistics for the traffic ips in the group

Returns Traffic IP statistics

Return type (dict)

3.3 virtual_servers

3.3.1 VirtualServers

class stingray.config.virtual_servers.**VirtualServers**(*host=None*, *port=None*, *user=None*, *password=None*, *api_version=None*, *ssl_verify=None*)

Bases: *stingray.apiclient.Client*

Class for interacting with Virtual Servers via the REST API

add(*server*, *pool*, *port*, ***server_props*)

Add a new virtual server.

Parameters

- **server** (*str*) – The virtual server name
- **pool** (*str*) – The default pool to use for traffic
- **port** (*int*) – The port to listen on
- **server_props** (*dict*) – Additional arguments to set the properties of the virtual server at time of creation. Must be a dict in the form of:

```
{'section': {'key': 'value'}}
```

Returns The new virtual server

Return type (*VirtualServer*)

delete(*server*)

Delete a virtual server

Parameters **server** (*str*) – The name of the virtual server to delete

Returns Response from the _api_delete method

Return type (dict)

classmethod from_client(*client*)

get(*server*)

Get a VirtualServer object for the requested virtual server

Parameters **server** (*str*) – The name of the virtual server to get

Returns The requested virtual server

Return type (*VirtualServer*)

3.3.2 VirtualServer

class stingray.config.virtual_servers.**VirtualServer**(*server_name*, *server_path=None*, *server_properties=None*, *host=None*, *port=None*, *user=None*, *password=None*, *api_version=None*, *ssl_verify=None*)

Bases: *stingray.apiclient.Client*

Class for interacting with individual virtual servers via the REST API

classmethod from_client(*client*, *server_name*, *server_path=None*, *server_properties=None*)

statistics()

Get statistics for the virtual server

Returns Virtual server statistics

Return type (dict)

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index

T

U

V