
stingray Documentation

Release 0.1.0

Mark Troyer

Aug 15, 2018

Contents:

1	Installing and Using the <code>stingray</code> Module	3
1.1	Installation	3
1.2	Usage Examples	3
2	<code>stingray.apiclient</code>	7
2.1	Client	7
2.2	StatusAPI	7
3	<code>stingray.config</code>	9
3.1	pools	9
3.2	traffic_ip_groups	9
3.3	virtual_servers	9
4	Indices and tables	11

`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.

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]:
{'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]:
{'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]:
{'Node1': {'node': 'Node1', 'state': 'active'},
 'Node2': {'node': 'Node2', 'state': 'active'}}

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

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

Update arbitrary pool properties:

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

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

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

In [11]: pool.update()

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

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


CHAPTER 2

stingray.apiclient

2.1 Client

2.2 StatusAPI

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

3.1.2 Pool

3.2 traffic_ip_groups

3.2.1 TrafficIPGroups

3.2.2 TrafficIPGroup

3.3 virtual_servers

3.3.1 VirtualServers

3.3.2 VirtualServer

CHAPTER 4

Indices and tables

- `genindex`
- `modindex`
- `search`