exabgpctl Documentation

Release 19.06-1

Ahmet Demir

Jun 17, 2019

Contents

exabgpctl is wrapper arround ExaBGP function.

- Enable / Disable maintenance
- View processes and neigbours
- Check neighbours connectivity
- Check processes statuses

The latest stable version is available on PyPI.

```
pip install -U exabgpctl
```

Getting started exabgpctl's getting-started!

guide/index All detailed guide for exabgpctl.

API Reference The complete API documentation — the innards of documents, querysets and fields.

CHAPTER 1

Community

To get help with using exabgpctl, create an issue on GitHub issues.

CHAPTER 2

Contributing

Yes please! We are always looking for contributions, additions and improvements.

The source is available on GitHub and contributions are always encouraged. Contributions can be as simple as minor tweaks to this documentation, the website or the core.

To contribute, fork the project on GitHub and send a pull request.

CHAPTER 3

Changes

See the *Changelog* for a full list of changes to exabgpctl.

CHAPTER 4

Offline Reading

Download the docs in pdf or epub formats for offline reading.

4.1 Getting started

4.1.1 Installing exabgpctl

exabgpctl is available on PyPI, so you can use **`pip`**:

```
$ pip install exabgpctl
```

Alternatively, if you don't have setuptools installed, download it from PyPi and run

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

To use the bleeding-edge version of exabgpctl, you can get the source from GitHub and install it as above:

```
$ git clone git://github.com/ahmet2mir/exabgpctl
$ cd exabgpctl
$ python setup.py install
```

4.1.2 Configuration

exabgpctl will not use any "self" configuration, we will only read the real exabgp conf and extend features. By default it will read the file under `/etc/exabgp/exabgp.conf`. To override it, set `environment variable`

- `EXABGPCTL_CONF`: exabgp.conf path (default /etc/exabgp/exabgp.conf)
- `EXABGPCTL_STATE`: where state files should be stored (for process state command) (default /var/lib/exabgp/status)

All examples using here will use conf from `examples` folder.

4.1.3 Bash Autocompletion

```
$ eval "$(_EXABGPCTL_COMPLETE=source exabgpctl)"
```

See Click project

4.1.4 Output format

You could choose which output format you wan't, by default it will be *json*

```
$ exabgpctl -o, --output [flat|json|yaml]
```

Where *flat* is key/value output.

4.1.5 Process Status

Check all process statuses, exabgpctl will read state and run the healthcheck command defined in exabgp.conf

```
$ exabgpctl process status
{
"service1.exabgp.lan": {
    "state": "UP",
    "command_check": "/bin/true",
    "command": true,
    "state_path": "/var/lib/exabgp/status/service1.exabgp.lan"
}
...
```

4.1.6 Enable / Disable process maintenance

ExaBGP support a maintenance flag, if the file exists, the route will be unannounced.

Disable

```
$ exabgpctl process disable service1.exabgp.lan
True
$ exabgpctl process status
{
"service1.exabgp.lan": {
    "state": "DISABLED",
    "command_check": "/bin/true",
    "command": true,
    "state_path": "/var/lib/exabgp/status/service1.exabgp.lan"
}
...
```

Enable

```
$ exabgpctl process enable service1.exabgp.lan
True
$ exabgpctl process status
{
"service1.exabgp.lan": {
```

(continues on next page)

(continued from previous page)

```
    "state": "UP",
    "command_check": "/bin/true",
    "command": true,
    "state_path": "/var/lib/exabgp/status/service1.exabgp.lan"
}
...
```

4.1.7 List process

List all process (with any state)

```
$ exabgpctl process list
[
    "service1.exabgp.lan",
    "service2.exabgp.lan",
    "service3.exabgp.lan"
]
```

List only disabled (maintenance) process

```
$ exabgpctl process disable service1.exabgp.lan
True
$ exabgpctl process list -d
[
    "service1.exabgp.lan"
]
```

4.1.8 Change state

exabgp could update the state of the process using `--execute` flag in healthcheck. And set an environment variable with the current state.

You could use exabgctl to manage this state

```
$ STATE='DOWN' exabgpctl process state service1.exabgp.lan
DOWN
$ exabgpctl process status
{
"service1.exabgp.lan": {
    "state": "DOWN",
    "command_check": "/bin/true",
    "command": true,
    "state_path": "/var/lib/exabgp/status/service1.exabgp.lan"
}
```

4.1.9 Show process

Get process details

```
$ exabgpctl process show service1.exabgp.lan
{
    "consolidate": false,
```

(continues on next page)

(continued from previous page)

```
    "receive-keepalives": false,
    "receive-packets": false,
    "receive-opens": false,
    "receive-refresh": false,
    "receive-notifications": false,
    "neighbor-changes": false,
    "encoder": "text",
    "receive-parsed": false,
    "neighbor": "*",
    "receive-operational": false,
    "run": {
    ...
```

4.1.10 List neighbors

List all process (with any state)

```
$ exabgpctl neighbor list
[
    "192.168.0.1",
    "192.168.0.2"
]
```

4.1.11 Show neighbor

Get neighbor details

```
$ exabgpctl neighbor show 192.168.0.1
{
    "group_updates": false,
    "add_path": 0,
    "flush": true,
    "api": {},
    "connect": 0,
    "ttl": null,
    "peer_address": "192.168.0.1",
...
```

4.1.12 Status neighbor

Get neighbor statuses, it will try to connect to neighbor on port 179.

```
$ exabgpctl neighbor status
{
    "192.168.0.2": {
        "status": false,
        "status_addressport": [
            "192.168.0.2",
            179
        ]
    },
```

(continues on next page)

(continued from previous page)

```
    "192.168.0.1": {
        "status": false,
        "status_addressport": [
            "192.168.0.1",
            179
        ]
    }
}
```

4.2 API Reference

4.2.1 Controller

exabgpctl.view

exception exabgpctl.controller.**ExabgpCTLError**
Generic Error to catch from view

exabgpctl.controller.**config_load**()
ExaBGP config loader. Loader will use exabgp lib to load the config like exabgp did

Returns configuration with path, state, version, neighbors and processes.

Return type dict

Examples

```
>>> import os
>>> os.environ['EXABGPCTL_CONF'] = "/etc/exabgp/exabgp.conf"
>>> os.environ['EXABGPCTL_STATE'] = "/var/lib/exabgp/state"
>>> cfg = config_load()
>>> cfg
{
    'path': '/tmp/exabgp/exabgp.conf',
    'state': '/tmp/exabgp/state',
    'version': {
        'python': '3.7.1',
        'exabgp': '3.4.19',
        'os': 'Linux-4.4.0-138-generic-x86_64-with',
        'exabgpctl': '19.01-1'
    },
    'neighbors': [
        {
            'local_address': '192.168.1.1',
            'local_as': 12345,
            'name': '192.168.0.1',
            'peer_address': '192.168.0.1',
            'peer_as': 67890,
            'router_id': '192.168.1.1',
            ...
        },
        {
            'local_address': '192.168.1.1',
```

(continues on next page)

(continued from previous page)

```
            'local_as': 12345,
            'name': '192.168.0.1',
            'peer_address': '192.168.0.1',
            'peer_as': 67890,
            'router_id': '192.168.1.1',
            ...
        }
    ],
    'processes': [
    {
        'name': 'service1.exabgp.lan',
        'run': {
            'ip_dynamic': False,
            'disabled_execute': None,
            'sudo': False,
            'pid': None,
            'community': '11223:344',
            'withdraw_on_down': True,
            'execute': ['/usr/bin/exabgpctl process state service1...'],
            'name': 'service1.exabgp.lan',
            'interval': 5,
            'disable': '/tmp/exabgp/maintenance/service1.exabgp.lan',
            'command': '/bin/mycheck',
            'timeout': 5,
            ...
        },
        ...
    },
    {
        'name': 'service2.exabgp.lan',
        'run': {
            'ip_dynamic': False,
            'disabled_execute': None,
            'sudo': False,
            'pid': None,
            'community': '11223:355',
            'withdraw_on_down': True,
            'execute': ['/usr/bin/exabgpctl process state service2...'],
            'name': 'service2.exabgp.lan',
            'interval': 5,
            'disable': '/tmp/exabgp/maintenance/service2.exabgp.lan',
            'command': '/bin/mycheck',
            'timeout': 5,
            ...
        },
        ...
    },
    {
        'name': 'service3.exabgp.lan',
        'run': {
            'ip_dynamic': False,
            'disabled_execute': None,
            'sudo': False,
            'pid': None,
            'community': '11223:366',
            'withdraw_on_down': True,
            'execute': ['/usr/bin/exabgpctl process state service3...'],
```

(continues on next page)

(continued from previous page)

```
            'name': 'service3.exabgp.lan',
            'interval': 5,
            'disable': '/tmp/exabgp/maintenance/service3.exabgp.lan',
            'command': '/bin/mycheck',
            'timeout': 5,
            ...
        },
        ...
    }
    ]
}
```

Raises *ExabgpCTLError* – if the conf file doesn't exists.

See also:

github.com/Exa-Networks/exabgp/qa/tests/parsing_test.py

`exabgpctl.controller.`**`disable_process`**(*cfg*, *process*)

Disable process (ie create maintenance file).

Parameters

- **cfg** (*dict*) – config from config_load.
- **process** (*str*) – process to disable.

Returns True if the file exists.

Return type bool

Examples

```
>>> disable_process(cfg, 'service1.exabgp.lan')
True
>>> list_disabled_processes(cfg)
['service1.exabgp.lan']
```

`exabgpctl.controller.`**`enable_process`**(*cfg*, *process*)

Enable process (ie create maintenance file).

Parameters

- **cfg** (*dict*) – config from config_load.
- **process** (*str*) – process to enable.

Returns True if the file exists.

Return type bool

Examples

```
>>> list_disabled_processes(cfg)
['service1.exabgp.lan']
>>> enable_process(cfg, 'service1.exabgp.lan')
True
```

(continues on next page)

(continued from previous page)

```
>>> list_disabled_processes(cfg)
[]
```

exabgpctl.controller.**flat**(*data*, *prefix=None*)

Flat the dict

Parameters

- **data** (*dict*) – the dict to flat. Must be a key/value dict.
- **prefix** (*str, optional*) – prefix key with a str value, defaults is None.

Returns flatted dict

Return type dict

Examples

```
>>> data = {
        "key1": {
            "key11": {
                "key111": "value111"
            },
            "key12": {
                "key121": "value121"
            }
        },
        "key2": ["one","two", "three"]
    }
>>> flat(data)
{
    'key1_key11_key111': 'value111',
    'key1_key12_key121': 'value121',
    'key2[1]': 'two',
    'key2[2]': 'three',
    'key2[0]': 'one'
}
```

See also:

- github.com/ahmet2mir/python-snippets/snippets/flat_unflat_dict.py

exabgpctl.controller.**get_neighbor**(*cfg*, *name*)

Show neighbor details.

Parameters

- **cfg** (*dict*) – config from config_load.
- **name** (*str*) – neighbor to retrieve.

Returns neighbor data.

Return type dict

Examples

```
>>> get_neighbor(cfg, '192.168.0.2')
{
    'local_address': '192.168.1.1',
    'local_as': 12345,
    'name': '192.168.0.1',
    'peer_address': '192.168.0.1',
    'peer_as': 67890,
    'router_id': '192.168.1.1',
    ...
}
```

Raises *ExabgpCTLError* – If neighbor not found.

exabgpctl.controller.**get_process**(*cfg*, *name*)
Show process details.

Parameters

- **cfg** (*dict*) – config from config_load.
- **name** (*str*) – process to retrieve.

Returns process data.

Return type dict

Examples

```
>>> get_process(cfg, 'service1.exabgp.lan')
{
    'name': 'service1.exabgp.lan',
    'run': {
        'ip_dynamic': False,
        'disabled_execute': None,
        'sudo': False,
        'pid': None,
        'community': '11223:344',
        'withdraw_on_down': True,
        'name': 'service1.exabgp.lan',
        'interval': 5,
        'disable': '/tmp/exabgp/maintenance/service1.exabgp.lan',
        'command': '/bin/mycheck',
        'timeout': 5,
        ...
    },
    ...
}
```

Raises *ExabgpCTLError* – If process not found.

exabgpctl.controller.**get_version**(*key=None*)
Get module, deps and platform version informations.

Parameters **key** (*str*) – filter item

Returns

With exabgp, exabgctl, python and os versions. If key specified will return a string.

Return type dict

Examples

```
>>> get_version()
{
    'python': '3.7.1',
    'exabgp': '3.4.19',
    'os': 'Linux-4.4.0-138-generic-x86_64-with',
    'exabgpctl': '19.01-1'
}
>>> get_version("exabgpctl")
'19.01-1'
```

exabgpctl.controller.**list_disabled_processes**(*cfg*)

List disabled processes from config.

Parameters **cfg** (*dict*) – config from config_load.

Returns list of string with process names.

Return type list

Examples

```
>>> list_disabled_processes(cfg)
['service1.exabgp.lan']
```

exabgpctl.controller.**list_enabled_processes**(*cfg*)

List enabled processes from config.

Parameters **cfg** (*dict*) – config from config_load.

Returns list of string with process names.

Return type list

Examples

```
>>> list_enabled_processes(cfg)
['service2.exabgp.lan', 'service3.exabgp.lan']
```

exabgpctl.controller.**list_neighbors**(*cfg*)

List neighbors from config.

Parameters **cfg** (*dict*) – config from config_load.

Returns list of string with neighbor names.

Return type list

Examples

```
>>> list_neighbors(cfg)
['192.168.0.2', '192.168.0.1']
```

exabgpctl.controller.**list_processes**(*cfg*)
List processes from config.

Parameters **cfg** (*dict*) – config from config_load.

Returns list of string with process names.

Return type list

Examples

```
>>> list_processes(cfg)
['service1.exabgp.lan', 'service2.exabgp.lan', 'service3.exabgp.lan']
```

exabgpctl.controller.**print_flat**(*data*)
Print data in flat mode.

If data is not hash or list, will only print raw value.

Parameters **data** (*dict*) – data to print.

Examples

```
>>> data = {
        "key1": {
            "key11": {
                "key111": "value111"
            },
            "key12": {
                "key121": "value121"
            }
        },
        "key2": ["one","two", "three"]
    }
>>> print_flat(data)
key1__key11__key111=value111
key1__key12__key121=value121
key2[0]=one
key2[1]=two
key2[2]=three
```

exabgpctl.controller.**print_json**(*data*)
Print data in json mode.

If data is not hash or list, will only print raw value.

Parameters **data** (*dict*) – data to print.

Examples

```
>>> data = {
        "key1": {
            "key11": {
                "key111": "value111"
            },
            "key12": {
                "key121": "value121"
            }
        },
        "key2": ["one","two", "three"]
    }
>>> print_json(data)
{
    "key2": [
        "one",
        "two",
        "three"
    ],
    "key1": {
        "key12": {
            "key121": "value121"
        },
        "key11": {
            "key111": "value111"
        }
    }
}
```

exabgpctl.controller.**print_yaml**(*data*)

Print data in yaml mode.

If data is not hash or list, will only print raw value.

Parameters **data** (*dict*) – data to print.

Examples

```
>>> data = {
        "key1": {
            "key11": {
                "key111": "value111"
            },
            "key12": {
                "key121": "value121"
            }
        },
        "key2": ["one","two", "three"]
    }
>>> print_yaml(data)
---
key1:
  key11:
    key111: value111
  key12:
    key121: value121
```

(continues on next page)

(continued from previous page)

```
key2:
  - one
  - two
  - three
```

exabgpctl.controller.**state_process**(*cfg*, *process*)

Set exabgp state in a statefile.

ExaBGP healthcheck command could run an action on each state change using environment called "STATE". See healthcheck –execute option.

Parameters

- **cfg** (*dict*) – config from config_load.
- **process** (*str*) – process to enable.

Returns

One of exabgp stat INIT: Initial state DISABLED: Disabled state RISING: Checks are currently succeeding. FALLING: Checks are currently failing. UP: Service is considered as up. DOWN: Service is considered as down.

Return type str

Examples

```
>>> import os
>>> os.environ["STATE"] = "UP"
>>> state_process(cfg, 'service1.exabgp.lan')
'UP'
>>> with open(cfg["state"] + "/service1.exabgp.lan", "r") as fd:
...     fd.read()
'UP'
```

exabgpctl.controller.**status_neighbors**(*cfg*)

Check connectivity with neighbors.

Parameters **cfg** (*dict*) – config from config_load.

Returns with statuses for each neighbor.

Return type dict

Examples

```
>>> status_neighbors(cfg)
{
    '192.168.0.1': {
        'status': True,
        'status_addressport': ['192.168.0.1', 179]
    },
    '192.168.0.2': {
        'status': True,
        'status_addressport': ['192.168.0.2', 179]
    }
}
```

exabgpctl.controller.**status_processes**(*cfg*)
Read all states from statefiles and run using healthcheck commands.

Parameters

- **cfg** (*dict*) – config from config_load.
- **process** (*str*) – process to enable.

Returns with statuses for each process.

Return type dict

Examples

```
>>> status_processes(cfg)
{
    'service1.exabgp.lan': {
        'state': 'UP',
        'state_path': '/tmp/exabgp/state/service1.exabgp.lan',
        'command': True,
        'command_check': '/bin/mycheck'
    },
    'service2.exabgp.lan': {
        'state': 'DOWN',
        'state_path': '/tmp/exabgp/state/service2.exabgp.lan',
        'command': False,
        'command_check': '/bin/mycheck'
    },
    'service3.exabgp.lan': {
        'state': 'DOWN',
        'state_path': '/tmp/exabgp/state/service3.exabgp.lan',
        'command': False,
        'command_check': '/bin/mycheck'
    }
}
```

exabgpctl.controller.**tcping**(*address*, *port*)
Like tcping tools, will test if the address:port is open.

Parameters

- **address** (*str*) – target address ip.
- **port** (*int*) – target port.

Returns True if address:port is open.

Return type bool

Examples

```
>>> tcping('8.8.8.8', 53)
True
```

4.3 Changelog

4.3.1 Changes in 19.01-1

- First release

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

Index