python-ovs-vsctl Documentation

Release 2.6.0.0

Iwase Yusuke

Jul 12, 2020

Contents

`python-ovs-vsctl` provides objects for OVSDB requests by wrapping `ovs-vsctl` command which included in Open vSwitch.

Internally, `python-ovs-vsctl` does this by just invoking `ovs-vsctl` command.

CHAPTER 1

Getting Started

1.1 Installation

Currently, only installing from GitHub via pip is supported.

```
$ pip install git+https://github.com/iwaseyusuke/python-ovs-vsctl.git
```

1.2 Basic Usage

`python-ovs-vsctl` sends OVSDB requests via TCP or SSH connections to switches. Please make sure that the manager targets are configured on the switches before calling the APIs of `python-ovs-vsctl` as following.

```
$ sudo ovs-vsctl set-manager ptcp:6640
```

Note: The port 6640 is the IANA registered port for OVSDB.

Then, let's call the APIs for OVSDB.

```
>>> from ovs_vsctl import VSCtl
>>> vsctl = VSCtl('tcp', '127.0.0.1', 6640)
>>> vsctl.run(command='show')
<subprocess.Popen object at 0x7fd4e86688d0>
```

Also, you can print the outputs like `ovs-vsctl` command.

```
>>> popen = vsctl.run('show')
>>> print(popen.stdout.read())
77a5cb2b-6a99-449b-adbe-19cfc41ef103
    Manager "ptcp:6640"
```

(continues on next page)

(continued from previous page)

```
    Bridge "s1"
        Controller "ptcp:6634"
        Controller "tcp:127.0.0.1:6633"
        fail_mode: secure
        Port "s1-eth1"
            Interface "s1-eth1"
        Port "s1"
            Interface "s1"
                type: internal
        Port "s1-eth2"
            Interface "s1-eth2"
    ovs_version: "2.5.0"
```

But, this format is not convenient for Python programming... If we can get the outputs as json loaded object, it is more useful, isn't it?

To parse the outputs, you can use the `ovs-vsctl` command parsers.

```
>>> from ovs_vsctl import list_cmd_parser
>>> vsctl.run('list port', parser=list_cmd_parser)
[Record(_uuid='91c8423c-f032-4e0f-a3e2-9e80bddcd5aa', bond_active_slave=[], bond_
↪downdelay=0, bond_fake_iface=False, bond_mode=[], bond_updelay=0, external_ids={},␣
↪fake_bridge=False, interfaces='59a88084-fb01-4d0f-b413-0905336e5957', lacp=[],␣
↪mac=[], name='s1-eth1', other_config={}, qos=[], rstp_statistics={}, rstp_status={},
↪ statistics={}, status={}, tag=[], trunks=[], vlan_mode=[]), Record(_uuid='99f9d4a5-
↪948d-4ed6-9b4c-4d64ada88e5a', bond_active_slave=[], bond_downdelay=0, bond_fake_
↪iface=False, bond_mode=[], bond_updelay=0, external_ids={}, fake_bridge=False,␣
↪interfaces='8df33dba-cb3b-4d72-891a-9591f2d1e115', lacp=[], mac=[], name='s1',␣
↪other_config={}, qos=[], rstp_statistics={}, rstp_status={}, statistics={}, status=
↪{}, tag=[], trunks=[], vlan_mode=[]), Record(_uuid='ba7fee67-2e97-470a-9df5-
↪446d72fa1645', bond_active_slave=[], bond_downdelay=0, bond_fake_iface=False, bond_
↪mode=[], bond_updelay=0, external_ids={}, fake_bridge=False, interfaces='bab7f26e-
↪be1e-431c-bd99-27fed50c94c5', lacp=[], mac=[], name='s1-eth2', other_config={},␣
↪qos=[], rstp_statistics={}, rstp_status={}, statistics={}, status={}, tag=[],␣
↪trunks=[], vlan_mode=[])]
```

`list_cmd_parser` parses the outputs into the list of `Record` object which contain the information of record of OVSDB table.

For more details of the APIs of `python-ovs-vsctl`, please refer to the API Reference of this documentation.

CHAPTER 2

API Reference

2.1 ovs_vsctl.__init__

Slightly ovs-vsctl wrapper for Python.

class ovs_vsctl.**VSCtl**(*protocol='tcp'*, *addr='127.0.0.1'*, *port=6640*)
Runner class for 'ovs-vsctl' command.

Parameters

- **protocol** – *'tcp'*, *'ssl'*, and *'unix'* are available.
- **addr** – IP address of switch to connect.
- **port** – (TCP or SSL) port number to connect.

Raise

- ValueError – When the given parameter is invalid.

ovsdb_addr
Returns OVSDB server address formatted like '–db' option of 'ovs-vsctl' command.

Example:

```
>>> from ovs_vsctl import VSCtl
>>> vsctl = VSCtl('tcp', '127.0.0.1', 6640)
>>> vsctl.ovsdb_addr
'tcp:127.0.0.1:6640'
```

Returns OVSDB server address.

run(*command*, *table_format='list'*, *data_format='string'*, *parser=None*)
Executes ovs-vsctl command.

command is an str type and the format is the same as 'ovs-vsctl' except for omitting 'ovs-vsctl' in command format.

For example, if you want to get the list of ports, the command for 'ovs-vsctl' should like 'ovs-vsctl list port' and *command* argument for this method should be:

```
>>> from ovs_vsctl import VSCtl
>>> vsctl = VSCtl('tcp', '127.0.0.1', 6640)
>>> vsctl.run(command='list port')
<subprocess.Popen object at 0x7fbbe9d549e8>
```

Parameters

- **command** – Command to execute.
- **table_format** – Table format. Meaning is the same as '–format' option of 'ovs-vsctl' command.
- **data_format** – Cell format in table. Meaning is the same as '–data' option of 'ovs-vsctl' command.
- **parser** – Parser class for the outputs. If this parameter is specified *table_format* and *data_format* is overridden with *table_format='list'* and *data_format='json'*.

Returns Output of 'ovs-vsctl' command. If *parser* is not specified, returns an instance of 'subprocess.Popen'. If *parser* is specified, the given *parser* is applied to parse the outputs.

Raise

- ovs_vsctl.exception.VSCtlCmdExecError – When the given command fails.
- ovs_vsctl.exception.VSCtlCmdParseError – When the given parser fails to parse the outputs.

ovs_vsctl.**line_parser**(*buf*)

Parses the given *buf* as str representation of list of values (e.g. 'ovs-vsctl list-br' command).

Parameters **buf** – str type value containing values list.

Returns list of parsed values.

ovs_vsctl.**list_cmd_parser**(*buf*)

Parser for 'ovs-vsctl list' and 'ovs-vsctl find' command.

buf must be the str type and the output of 'ovs-vsctl list' or ovs-vsctl find' command with '–format=list' and '–data=json' options.

Parameters **buf** – str type output of 'ovs-vsctl list' command.

Returns list of *Record* instances.

ovs_vsctl.**find_cmd_parser**(*buf*)

Parser for 'ovs-vsctl list' and 'ovs-vsctl find' command.

buf must be the str type and the output of 'ovs-vsctl list' or ovs-vsctl find' command with '–format=list' and '–data=json' options.

Parameters **buf** – str type output of 'ovs-vsctl list' command.

Returns list of *Record* instances.

ovs_vsctl.**get_cmd_parser**(*buf*)

Parser for 'ovs-vsctl get' command.

buf must be the str type and the output of 'ovs-vsctl get' command.

Assumption: The output is mostly formatted in json, except for 'uuid' and 'key' of map type value.

Parameters **buf** – value of 'ovs-vsctl get' command.

Returns python object corresponding to the value type of row.

2.2 ovs_vsctl.vsctl

APIs for execute 'ovs-vsctl' command.

class ovs_vsctl.vsctl.**VSCtl**(*protocol='tcp'*, *addr='127.0.0.1'*, *port=6640*)
Runner class for 'ovs-vsctl' command.

Parameters

- **protocol** – *'tcp'*, *'ssl'*, and *'unix'* are available.
- **addr** – IP address of switch to connect.
- **port** – (TCP or SSL) port number to connect.

Raise

- ValueError – When the given parameter is invalid.

ovsdb_addr
Returns OVSDB server address formatted like '–db' option of 'ovs-vsctl' command.

Example:

```
>>> from ovs_vsctl import VSCtl
>>> vsctl = VSCtl('tcp', '127.0.0.1', 6640)
>>> vsctl.ovsdb_addr
'tcp:127.0.0.1:6640'
```

Returns OVSDB server address.

run(*command*, *table_format='list'*, *data_format='string'*, *parser=None*)
Executes ovs-vsctl command.

command is an str type and the format is the same as 'ovs-vsctl' except for omitting 'ovs-vsctl' in command format.

For example, if you want to get the list of ports, the command for 'ovs-vsctl' should like 'ovs-vsctl list port' and *command* argument for this method should be:

```
>>> from ovs_vsctl import VSCtl
>>> vsctl = VSCtl('tcp', '127.0.0.1', 6640)
>>> vsctl.run(command='list port')
<subprocess.Popen object at 0x7fbbe9d549e8>
```

Parameters

- **command** – Command to execute.
- **table_format** – Table format. Meaning is the same as '–format' option of 'ovs-vsctl' command.
- **data_format** – Cell format in table. Meaning is the same as '–data' option of 'ovs-vsctl' command.

- **parser** – Parser class for the outputs. If this parameter is specified *table_format* and *data_format* is overridden with *table_format='list'* and *data_format='json'*.

Returns Output of 'ovs-vsctl' command. If *parser* is not specified, returns an instance of 'subprocess.Popen'. If *parser* is specified, the given *parser* is applied to parse the outputs.

Raise

- ovs_vsctl.exception.VSCtlCmdExecError – When the given command fails.
- ovs_vsctl.exception.VSCtlCmdParseError – When the given parser fails to parse the outputs.

2.3 ovs_vsctl.parser

Parsers for 'ovs-vsctl' command outputs.

class ovs_vsctl.parser.**Record**(***kwargs*)
Record object of OVSDB table.

Attributes are corresponding to columns of parsed tables.

classmethod parse(*buf*)
Parses the given *buf* as str containing a record of rows.

Parameters **buf** – Record in str type.

Returns *Record* instance.

ovs_vsctl.parser.**find_cmd_parser**(*buf*)
Parser for 'ovs-vsctl list' and 'ovs-vsctl find' command.

buf must be the str type and the output of 'ovs-vsctl list' or ovs-vsctl find' command with '–format=list' and '–data=json' options.

Parameters **buf** – str type output of 'ovs-vsctl list' command.

Returns list of *Record* instances.

ovs_vsctl.parser.**get_cmd_parser**(*buf*)
Parser for 'ovs-vsctl get' command.

buf must be the str type and the output of 'ovs-vsctl get' command.

Assumption: The output is mostly formatted in json, except for 'uuid' and 'key' of map type value.

Parameters **buf** – value of 'ovs-vsctl get' command.

Returns python object corresponding to the value type of row.

ovs_vsctl.parser.**line_parser**(*buf*)
Parses the given *buf* as str representation of list of values (e.g. 'ovs-vsctl list-br' command).

Parameters **buf** – str type value containing values list.

Returns list of parsed values.

ovs_vsctl.parser.**list_cmd_parser**(*buf*)
Parser for 'ovs-vsctl list' and 'ovs-vsctl find' command.

buf must be the str type and the output of 'ovs-vsctl list' or ovs-vsctl find' command with '–format=list' and '–data=json' options.

Parameters **buf** – str type output of 'ovs-vsctl list' command.

Returns list of *Record* instances.

ovs_vsctl.parser.**show_cmd_parser**(*buf*)
Parser for 'ovs-vsctl show' command.

Currently, parses ONLY 'ovs_version' column.

Parameters **buf** – str type output of 'ovs-vsctl show' command.

Returns dict type value of 'ovs-vsctl show' command.

2.4 ovs_vsctl.exception

Exception classes.

exception ovs_vsctl.exception.**VSCtlCmdExecError**
Raised when 'ovs-vsctl' command returns non-zero exit code.

exception ovs_vsctl.exception.**VSCtlCmdParseError**
Raised when user specified parser fails to parse the outputs of 'ovs-vsctl' command.

2.5 ovs_vsctl.utils

Utilities.

ovs_vsctl.utils.**is_valid_uuid**(*uuid*)
Returns *True* if the given *uuid* is valid, otherwise returns *False*.

Parameters **uuid** – str type value to be validated.

Returns *True* if valid, else *False*.

ovs_vsctl.utils.**run**(*args*)
Wrapper of 'subprocess.run'.

Parameters **args** – Command arguments to execute.

Returns instance of 'subprocess.Popen'.

Python Module Index

o

Index