mdssdk

Release 1.4.0

Aug 17, 2022

Contents:

CHAPTER 1

Python SDK/API library for Cisco MDS Switches.

This library will be useful for automating day to day tasks or developing new tools which involve Cisco MDS switches

- Python version: 3.6 and above
- Supports both NXAPI and SSH
- Limited support for N9K and FI
- Apache License, Version 2.0 (the "License")

1.1 Installation Steps

1.1.1 From pip:

Installs the latest version.

```
pip install mdssdk
export NET_TEXTFSM=$HOME/mdssdk-templates/
```

1.1.2 From github:

```
git clone https://github.com/Cisco-SAN/mdssdk.git
cd mdssdk
python setup.py install
pip install -r requirements.txt
export NET_TEXTFSM=$HOME/mdssdk-templates/
```

- `mdssdk` requires `NET_TEXTFSM` environment variable to be set
- This variable points to the directory where the textfsm templates are copied to
- To set the env please execute the below command after installing `mdssdk` ..

```
export NET_TEXTFSM=$HOME/mdssdk-templates/
```

- It is recommended that you add this env permanently into your `.bashrc` or `.cshrc` file

1.2 Uninstallation Steps

To uninstall mdssdk,

```
pip uninstall mdssdk
```

1.3 Documentation

- http://mdssdk.readthedocs.io

1.4 Support Matrix

NXOS Version	SDK Version
9.2(2) and below	v1.4.0
9.2(1) and below	v1.3.3
8.5(1) and below	v1.2.0
8.4(2b) and below	v1.1.0
8.4(2a) and below	v1.0.1

CHAPTER 2

Modules

2.1 Switch

class mdssdk.switch.**Switch**(*ip_address*, *username*, *password=None*, *connection_type='https'*, *ssh_key_file=None*, *port=None*, *timeout=100*, *verify_ssl=True*)

Switch module

Parameters

- **ip_address** (*str*) – mgmt ip address of switch
- **username** – username
- **password** (*str*) – password (optional for ssh keys)
- **connection_type** (*str*) – connection type 'http' or 'https' or 'ssh' (default: 'https')
- **ssh_key_file** (*str*) – file name of SSH key file (optional for password auth)
- **port** (*int*) – port number (default: 8443 for https and 8080 for http) , ignored when connection type is ssh
- **timeout** (*int*) – timeout period in seconds (default: 30)
- **verify_ssl** (*bool*) – SSL verification (default: True)

Example

```
>>> switch_obj = Switch(ip_address = switch_ip, username = switch_
↪username, password = switch_password)
>>> # For auth with ssh key file
>>> switch_obj = Switch(ip_address = switch_ip, username = switch_
↪username, connection_type = "ssh", ssh_key_file = './ssh/test_rsa'))
```

config(*command*, *rpc='2.0'*, *method='cli'*, *use_ssh=False*, *timeout=100*)

Send any command to run from the config mode

Parameters **command** (*str*) – command to send to the switch

Raises **CLIError** – If there is a problem with the supplied command.

Returns command output

discover_peer_switches()

Returns list of switch ips discovered

form_factor

Returns the form factor of the switch, i.e if its a 10 slot or 6 slot or 1RU or 2RU etc..

Returns Returns form factor of the switch or returns None if form factor could not be fetched from the switch

Return type str

Example

```
>>> print(switch_obj.form_factor)
10 slot
>>>
>>> print(switch2_obj.form_factor)
2 RU
>>>
```

image_string

Returns the image's string that is specific to a particular platform example m9700-sf3ek9, m9100-s6ek9 etc..

Returns Returns image string of the switch or returns None if image string could not be fetched from the switch

Return type str

Example

```
>>> print(switch_obj.image_string)
m9700-sf3ek9
>>>
>>> print(switch2_obj.image_string)
m9300-s2ek9
>>>
```

ipaddr

Get mgmt IPv4 address of the switch

Returns IPv4 address of switch

Return type str

Example

```
>>> print(switch_obj.ipaddr)
10.126.94.101
>>>
```

kickstart_image

Returns the kickstart image of the switch

Returns Returns kickstart image of the switch or returns None if kickstart image could not be fetched from the switch

Return type str

Example

```
>>> print(switch_obj.kickstart_image)
bootflash:///m9700-sf3ek9-kickstart-mz.8.4.1.bin
>>>
>>> print(switch2_obj.kickstart_image)
bootflash:///m9300-s2ek9-kickstart-mz.8.4.1.bin
>>>
```

last_boot_time

Returns the last boot time of the switch

Returns Returns the last boot time of the switch

Return type datetime.datetime

Example

```
>>> print(switch_obj.last_boot_time)
datetime.datetime(2021, 6, 15, 11, 14, 51, 617398)
>>>
```

model

Returns model of the switch

Returns Returns model of the switch or returns None if model could not be fetched from the switch

Return type str

Example

```
>>> print(switch_obj.model)
MDS 9710 (10 Slot) Chassis
>>>
>>> print(switch2_obj.model)
MDS 9396T 96X32G FC (2 RU) Chassis
>>>
```

name

get switchname or set switchname

Getter

Returns switch name

Return type str

Example

```
>>> print(switch_obj.name)
swTest
>>>
```

Setter

Parameters **name** (*str*) – name of the switch that needs to be set

Example

```
>>> switch_obj.name = "yourswitchname"
>>>
```

> **Warning:** Switch name must start with a letter, end with alphanumeric and contain alphanumeric and hyphen only. Max size 32.

npv
Check if switch is in NPV mode

Returns Returns True if switch is in NPV, else returns False

Return type bool

Example

```
>>> print(switch_obj.npv)
False
>>>
```

product_id
Get mgmt product_id address of the switch

Returns product_id address of switch

Return type str

Example

```
>>> print(switch_obj.product_id)
DS-C9706
>>>
```

serial_num
Get serial number of the switch

Returns serial number of switch

Return type str

Example

```
>>> print(switch_obj.serial_num)
FXS1928Q402
>>>
```

show(*command*, *raw_text=False*, *use_ssh=False*, *expect_string=None*, *timeout=100*)
Send a show command to the switch

Parameters

- **command** (*str*) – The command to send to the switch.
- **raw_text** (*bool (default: False)*) – If true then returns the command output in raw text(str) else it returns structured data(dict)
- **use_ssh** (*bool (default: False)*) – If true then the cmd is sent over ssh channel
- **expect_string** (*str (default: None)*) – string to expect after sending the show cmd, if set to None then it will expect the default string which is the cmd prompt
- **timeout** (*int (default: 100)*) – timeout for the show cmd sent

Raises **CLIError** – If there is a problem with the supplied command.

Returns The output of the show command, which could be raw text(str) or structured data(dict).

Return type dict

system_image
Returns the switch image of the switch

Returns Returns switch image of the switch or returns None if switch image could not be fetched from the switch

Return type str

Example

```
>>> print(switch_obj.system_image)
bootflash:///m9700-sf3ek9-mz.8.4.1.bin
>>>
>>> print(switch2_obj.system_image)
bootflash:///m9300-s2ek9-mz.8.4.1.bin
>>>
```

system_uptime
Returns the switch uptime

Returns Returns the switch uptime

Return type datetime.timedelta

Example

```
>>> print(switch_obj.system_uptime)
datetime.timedelta(days=7, seconds=7561)
>>>
```

type
Returns the type of the switch, i.e if its a 9710 or 9706 or 9396T etc..

Returns Returns type of the switch or returns None if type could not be fetched from the switch

Return type str

Example

```
>>> print(switch_obj.type)
9710
>>>
>>> print(switch2_obj.type)
9396T
>>>
```

version
Get the switch software version

Returns version

Return type str

Raises **CLIError** – Raises if there was a command error or some generic error due to which version could not be fetched

Example

```
>>> print(switch_obj.version)
8.4(2)
>>>
```

2.2 Module

class mdssdk.module.**Module**(*switch*, *mod_num*, *modinfo*)

Switch's module class

Example

```
>>> switch_obj = Switch(ip_address = switch_ip, username = switch_
↪username, password = switch_password )
>>> mod_handler = switch_obj.modules
>>> print(mod_handler)
[{1: <mdslib.module.Module object at 0x10ad710d0>}, {2: <mdslib.
↪module.Module object at 0x10ad71190>},
{3: <mdslib.module.Module object at 0x10ad711d0>}, {4: <mdslib.module.
↪Module object at 0x10ad71050>},
{5: <mdslib.module.Module object at 0x10abdf190>}]
```

model

Get model of the module

Returns model of the module

Return type str

Example

```
>>> switch_obj = Switch(ip_address = switch_ip, username = switch_
↪username, password = switch_password )
>>> mod_handler = list(switch_obj.modules.values())
>>> first_mod_handler = mod_handler[0]
>>> print(first_mod_handler.model)
DS-X9448-768K9
>>>
```

module_number

Get module number

Returns module number

Return type int

Example

```
>>> switch_obj = Switch(ip_address = switch_ip, username = switch_
↪username, password = switch_password )
>>> mod_handler = list(switch_obj.modules.values())
>>> first_mod_handler = mod_handler[0]
>>> print(first_mod_handler.module_number)
2
>>>
```

ports

Get number of ports on the module

Returns number of ports on the module

Return type int

Example

```
>>> switch_obj = Switch(ip_address = switch_ip, username = switch_
↪username, password = switch_password )
>>> mod_handler = list(switch_obj.modules.values())
>>> first_mod_handler = mod_handler[0]
>>> print(first_mod_handler.ports)
48
>>>
```

status
Get status of the module

Returns status of the module

Return type str

Example

```
>>> switch_obj = Switch(ip_address = switch_ip, username = switch_
↪username, password = switch_password )
>>> mod_handler = list(switch_obj.modules.values())
>>> first_mod_handler = mod_handler[0]
>>> print(first_mod_handler.status)
ok
>>>
```

type
Get type of the module

Returns type of the module

Return type str

Example

```
>>> switch_obj = Switch(ip_address = switch_ip, username = switch_
↪username, password = switch_password )
>>> mod_handler = list(switch_obj.modules.values())
>>> first_mod_handler = mod_handler[0]
>>> print(first_mod_handler.type)
2/4/8/10/16 Gbps Advanced FC Module
>>>
```

2.3 Vsan

class mdssdk.vsan.**Vsan**(*switch*, *id*)
Vsan module

Parameters

- **switch** (Switch) – switch object on which vsan operations need to be executed
- **id** (*int*) – vsan id

Example

```
>>> vsan_obj = Vsan(switch=switch_obj, id=2)
```

Warning: id must be within range 1-4094 (4079,4094 are reserved)

add_interfaces(*interfaces*)
Add interfaces to the vsan

Parameters **interfaces** (*list(Fc or PortChannel)*) – interfaces to be added to the vsan

Raises

- **VsanNotPresent** – if vsan is not present on the switch
- **InvalidInterface** – if the interface is not among supported interface types ('fc' and 'port-channel')
- **CLIError** – if the switch raises a error for the cli command passed

Returns None

Example

```
>>> fc = Fc(switch,"fc1/1")
>>> pc = PortChannel(switch,1)
>>> vsan_obj.add_interfaces([fc,pc])
>>> vsan_obj.add_interfaces(fc)
Traceback (most recent call last):
...
TypeError: Fc object is not iterable
```

create(*name=None*)
Creates vsan on the switch

Parameters **name** (*str or None*) – name of vsan (optional parameter, defaults to 'VSAN<vsan-id>' if passed as None)

Returns None

Example

```
>>> vsan_obj.create("vsan_2")
```

delete()
Deletes the vsan on the switch

Param None

Returns None

Raises **VsanNotPresent** – if vsan is not present on the switch

Example

```
>>> vsan_obj.delete()
```

id
Get vsan id

Returns id of the vsan if vsan is present on the switch, otherwise returns None

Return type int

Range 1 to 4094

name

Get the name of the vsan or Set the name of the vsan

Getter

Returns name of the vsan, returns None if vsan is not present on the switch

Return type str

Example

```
>>> print(vsan_obj.name)
"VSAN0001"
>>>
```

Setter

Parameters **name** (*str*) – name of the vsan

Example

```
>>> vsan_obj.name = "vsan_2"
```

state

Get the state of the vsan

Returns state of the vsan returns None if vsan is not present on the switch

Values return values are either 'active' or 'suspended'

suspend

Set the state of the vsan

Setter

Parameters **value** (*bool*) – if true suspends the vsan, else does a 'no suspend'

Raises **TypeError** – If the passed value is not of type bool

Example

```
>>> vsan_obj.suspend = True
```

2.4 DeviceAlias

class mdssdk.devicealias.**DeviceAlias**(*switch*)

Device Alias module

Parameters **switch** (Switch) – switch object on which device-alias operations needs to be executed

Example

```
>>> da = DeviceAlias(switch = switch_obj)
```

clear_database()

Clears database entries

Param None

Returns None

Raises **CLIError** – If there is any cli command error

Example

```
>>>
>>> da = DeviceAlias(switch = switch_obj)
>>> da.clear_database()
>>>
```

clear_lock()

Clears lock if lock is acquired

Param None

Returns None

Example

```
>>>
>>> da = DeviceAlias(switch = switch_obj)
>>> da.clear_lock()
>>>
```

create(*namepwwn*)

Create device alias entries

Parameters **namepwwn** (*dict (name: pwwn)*) – name and pwwn

Returns None

Raises **CLIError** – If there is any cli command error

Example

```
>>>
>>> da = DeviceAlias(switch = switch_obj)
>>> da.create({'device1': '21:00:00:0e:1e:30:34:a5','device2':
↪'21:00:00:0e:1e:30:3c:c5'})
>>>
```

database

Returns device-alias database in dict(name:pwwn) format, if there are no device-alias entries then it returns None

Returns database or None

Return type dict(name:pwwn)

delete(*name*)

Delete device alias entry

Parameters **name** (*str*) – name of device alias that needs to be deleted

Returns None

Raises **CLIError** – If there is any cli command error

Example

```
>>>
>>> da = DeviceAlias(switch = switch_obj)
>>> da.delete('device1')
```

distribute
set device-alias distribute configuration or get device-alias distribute configuration

Getter

Returns distribute

Return type bool

Example

```
>>>
>>> da = DeviceAlias(switch = switch_obj)
>>> print(da.distribute)
True
>>>
```

Setter

Parameters **distribute** (*bool*) – set to True if distribute needs to be enabled or set to False if distribute needs to be disabled

Raises

- **CLIError** – If there is any cli command error
- **TypeError** – If the passed value is not of type bool

Example

```
>>>
>>> da = DeviceAlias(switch = switch_obj)
>>> da.distribute = True
>>>
```

locked
Check if device-alias has acquired lock or not

Returns locked: Returns True if device-alias lock is acquired else returns False

Return type bool

mode
set device-alias mode or get device-alias mode

Getter

Returns mode

Return type str

Values [‘basic’, ‘enhanced’]

Example

```
>>>
>>> da = DeviceAlias(switch = switch_obj)
>>> print(da.mode)
```

(continues on next page)

(continued from previous page)

```
enhanced
>>>
```

Setter

Parameters **mode** (*str*) – mode

Values ['basic', 'enhanced']

Raises

- **InvalidMode** – if mode is not to either 'basic' or 'enhanced'
- **CLIError** – If there is any cli error

Example

```
>>>
>>> da = DeviceAlias(switch = switch_obj)
>>> da.mode = 'basic'
>>>
```

rename (*oldname*, *newname*)
Rename device alias entry

Parameters

- **oldname** (*str*) – old device alias name
- **newname** (*str*) – new device alias name

Returns None

Raises **CLIError** – If there is any cli command error

Example

```
>>>
>>> da = DeviceAlias(switch = switch_obj)
>>> da.rename('device1','device_new')
>>>
```

2.5 Fc

class mdssdk.fc.**Fc** (*switch*, *name*)
Fc interface module

Parameters

- **switch** (Switch) – switch object
- **name** (*str*) – name of fc interface

Raises **InvalidInterface** – when interface name is incorrect

Example

```
>>> fcobj = Fc(switch = switch_obj, name = "fc1/1")
```

analytics_type
get analytics type on the fc interface or set analytics type on the fc interface

Getter

Returns analytics type on the interface, None if there are no analytics configs

Return type str

Example

```
>>> fcobj = Fc(switch = switch_obj, name = "fc1/1")
>>> print(fcobj.analytics_type)
scsi
>>>
```

Setter

Parameters **type** (*str*) – set analytics type on the fc interface

Values scsi/nvme/all/None . Setting the value to None will remove the analytics config on the interface

Example

```
>>> fcobj = Fc(switch = switch_obj, name = "fc1/1")
>>> fcobj.analytics_type = 'scsi'
scsi
>>>
```

counters

Returns handler for counters module, using which we could get various counter details of the interface

Returns counters handler

Return type *Counters*

Example

```
>>> intcounters = int_obj.counters
>>>
```

description

set description of the interface or get description of the interface

Getter

Returns description of the interface

Return type str

Example

```
>>>
>>> print(int_obj.description)
This is an ISL connected to sw2
>>>
```

Setter

Parameters **description** (*str*) – set description of the interface

Example

```
>>>
>>> int_obj.description = "This is an ISL connected to sw2"
>>>
```

mode
set interface mode or get interface mode

Getter

Returns interface mode

Return type str

Example

```
>>>
>>> print(int_obj.mode)
F
>>>
```

Setter

Parameters **mode** (*str*) – set mode of the interface

Example

```
>>>
>>> int_obj.mode = "F"
>>>
```

name
get name of the interface

Returns name of the interface

Return type str

Example

```
>>>
>>> print(int_obj.name)
fc1/1
>>>
```

out_of_service
set out-of-service configuration for the fc interface

Parameters **value** (*bool*) – set to True to enable out-of-service, False otherwise

Example

```
>>> fcobj = Fc(switch = switch_obj, name = "fc1/1")
>>> fcobj.out_of_service = True
>>>
```

speed
set speed of the interface or get speed of the interface

Getter

Returns speed of the interface

Return type int

Example

```
>>>
>>> print(int_obj.speed)
32000
>>>
```

Setter

Parameters **mode** (*int*) – set speed of the interface

Example

```
>>>
>>> int_obj.speed = 32000
>>>
```

status

set status of the interface or get status of the interface

Getter

Returns status of the interface

Return type str

Example

```
>>>
>>> print(int_obj.status)
trunking
>>>
```

Setter

Parameters **mode** (*str*) – set status of the interface

Values “shutdown”, “no shutdown”

Example

```
>>>
>>> int_obj.status = "no shutdown"
>>>
```

transceiver

Returns handler for transceiver module, using which we could do transceiver related operations

Returns transceiver handler

Return type Transceiver

Example

```
>>> fcobj = Fc(switch = switch_obj, name = "fc1/1")
>>> trans_handler = fcobj.transceiver
>>>
```

trunk

set trunk mode on the interface or get trunk mode on the interface

Getter

Returns trunk mode of the interface

Return type str

Example

```
>>>
>>> print(int_obj.trunk)
on
>>>
```

Setter

Parameters **mode** (*str*) – set trunk mode on the interface

Example

```
>>>
>>> int_obj.trunk = "on"
>>>
```

class mdssdk.interface.Interface.**Counters**(*intobj*)

brief

Get brief counters details of the interface

Returns brief: Returns brief counters details of the interface

Return type dict (name:value)

Example

```
>>>
>>> intcounters = int_obj.counters
>>> print(intcounters.brief)
{'input_rate': 0, 'frames_in': 14970, 'output_rate': 0, 'frames_out
↪': 14831}
>>>
```

clear()

Clear the counters on the interface

Returns None

Example

```
>>>
>>> intcounters = int_obj.counters
>>> intcounters.clear()
>>>
```

congestion_stats

Get congestion stats from the detailed counters of the interface

Returns congestion_stats: congestion stats from the detailed counters of the interface

Return type dict (name:value)

Example

```
>>>
>>> intcounters = int_obj.counters
>>> print(intcounters.congestion_stats)
```

(continues on next page)

(continued from previous page)

```
{'timeout_discards': 0, 'credit_loss': 0, 'bb_scs_resend': 0, 'bb_
↪scr_incr': 0, 'txwait': 0,
'tx_wait_unavbl_1s': 0, 'tx_wait_unavbl_1m': 0, 'tx_wait_unavbl_1hr
↪': 0, 'tx_wait_unavbl_72hr': 0,
'rx_b2b_credit_remain': 1, 'tx_b2b_credit_remain': 0, 'tx_b2b_low_
↪pri_cre': 0, 'rx_b2b_credits': 0, 'tx_b2b_credits': 0}
>>>
```

link_stats

Get link stats from the detailed counters of the interface

Returns link_stats: link stats from the detailed counters of the interface

Return type dict (name:value)

Example

```
>>>
>>> intcounters = int_obj.counters
>>> print(intcounters.link_stats)
{'link_failures': 2, 'sync_loss': 0, 'signal_loss': 0, 'prm_seq_pro_
↪err': 0, 'inv_trans_err': 0,
'inv_crc': 0, 'delim_err': 0, 'frag_frames_rcvd': 0, 'frames_eof_
↪abort': 0, 'unknown_class_frames_rcvd': 0,
'runt_frames': 0, 'jabber_frames': 0, 'too_long': 0, 'too_short': 0,
↪ 'fec_corrected': 0, 'fec_uncorrected': 0,
'rx_link_reset': 0, 'tx_link_reset': 0, 'rx_link_reset_resp': 4,
↪'tx_link_reset_resp': 2, 'rx_off_seq_err': 6,
'tx_off_seq_err': 8, 'rx_non_oper_seq': 3, 'tx_non_oper_seq': 2}
>>>
```

loop_stats

Get loop stats from the detailed counters of the interface

Returns loop_stats: loop stats from the detailed counters of the interface

Return type dict (name:value)

Example

```
>>>
>>> intcounters = int_obj.counters
>>> print(intcounters.loop_stats)
{'rx_f8_lip_seq_err': 0, 'tx_f8_lip_seq_err': 0, 'rx_non_f8_lip_seq_
↪err': 0, 'tx_non_f8_lip_seq_err': 0}
>>>
```

other_stats

Get other stats from the detailed counters of the interface

Returns other_stats: other stats from the detailed counters of the interface

Return type dict (name:value)

Example

```
>>>
>>> intcounters = int_obj.counters
>>> print(intcounters.other_stats)
```

(continues on next page)

(continued from previous page)

```
{'pg_acl_drops': 0, 'pg_fib_start': '1', 'pg_fib_end': '16', 'pg_
↪fib_drops': 0, 'pg_xbar_start': '1',
'pg_xbar_end': '16', 'pg_xbar_drops': 0, 'pg_other_drops': 0}
>>>
```

total_stats
Get total stats from the detailed counters of the interface

Returns total_stats: total stats from the detailed counters of the interface

Return type dict (name:value)

Example

```
>>>
>>> intcounters = int_obj.counters
>>> print(intcounters.total_stats)
{'rx_total_frames': 14970, 'tx_total_frames': 14831, 'rx_total_bytes
↪': 2235488, 'tx_total_bytes': 1733508, 'rx_total_multicast': 0,
'tx_total_multicast': 0, 'rx_total_broadcast': 0, 'tx_total_
↪broadcast': 0, 'rx_total_unicast': 14970, 'tx_total_unicast':␣
↪14831,
'rx_total_discard': 0, 'tx_total_discard': 0, 'rx_total_error': 0,
↪'tx_total_error': 0, 'rx_c_2_frames': 0, 'tx_c_2_frames': 0,
'rx_c_2_bytes': 0, 'tx_c_2_bytes': 0, 'rx_c_2_discards': 0, 'rx_c_2_
↪port_rjt_frames': 0, 'rx_c_3_frames': 14962, 'tx_c_3_frames':␣
↪14823,
'rx_c_3_bytes': 2235072, 'tx_c_3_bytes': 1733092, 'rx_c_3_discards
↪': 0, 'rx_c_f_frames': 8, 'tx_c_f_frames': 8, 'rx_c_f_bytes': 416,
'tx_c_f_bytes': 416, 'rx_c_f_discards': 0}
>>>
```

2.6 PortChannel

class mdssdk.portchannel.**PortChannel**(*switch*, *id*)
PortChannel interface module extends Interface module

Parameters

- **switch** (Switch) – switch object
- **id** (*int*) – id of port-channel interface

Raises **InvalidPortChannelRange** – when it is not within 1 to 256

Example

```
>>> pcobj = PortChannel(switch = switch_obj, id = 1)
```

add_members(*interfaces*)
Add Fc members to the port channel

Parameters **interfaces** (*list (Fc)*) – list of Fc interfaces to be added

Raises **PortChannelNotPresent** – if port channel is not present on switch

Example

```
>>> pcobj = PortChannel(switch = switch_obj, id = 1)
>>> pcobj.create()
>>> fc1 = Fc( switch = switch_obj, name = "fc1/1")
>>> fc2 = Fc( switch = switch_obj, name = "fc1/2")
>>> pcobj.add_members([fc1,fc2])
>>>
```

channel_mode
set or get the channel mode of the port-channel

Getter

Returns Returns the channel mode of the port-channel

Return type str

Example

```
>>> pcobj = PortChannel(switch = switch_obj, id = 1)
>>> print(pcobj.channel_mode)
active
>>>
```

Setter

Parameters **mode** (*str*) – mode to which port-channel mode needs to be set

Values 'on', 'active'

Raises

- **InvalidChanelMode** – if mode is not 'on' or 'active'
- **PortChannelNotPresent** – if port-channel is not present on the switch

Example

```
>>> pcobj = PortChannel(switch = switch_obj, id = 1)
>>> pcobj.channel_mode = 'active'
>>>
```

counters
Returns handler for counters module, using which we could get various counter details of the interface

Returns counters handler

Return type *Counters*

Example

```
>>> intcounters = int_obj.counters
>>>
```

create()
Creates port-channel on switch

Example

```
>>> pcobj = PortChannel(switch = switch_obj, id = 1)
>>> pcobj.create()
```

delete()
Deletes port-channel on switch

Example

```
>>> pcobj = PortChannel(switch = switch_obj, id = 1)
>>> pcobj.delete()
```

description

set description of the interface or get description of the interface

Getter

Returns description of the interface

Return type str

Example

```
>>>
>>> print(int_obj.description)
This is an ISL connected to sw2
>>>
```

Setter

Parameters **description** (*str*) – set description of the interface

Example

```
>>>
>>> int_obj.description = "This is an ISL connected to sw2"
>>>
```

id

Returns port-channel id

Returns id of port-channel

Return type int

Example

```
>>> pcobj = PortChannel(switch = switch_obj, id = 1)
>>> print(pcobj.id)
1
>>>
```

members

Get the members of the port-channel

Returns members of the port-channel in dictionary format

Return type dict(name: obj(*Fc*))

mode

set interface mode or get interface mode

Getter

Returns interface mode

Return type str

Example

```
>>>
>>> print(int_obj.mode)
F
>>>
```

Setter

Parameters **mode** (*str*) – set mode of the interface

Example

```
>>>
>>> int_obj.mode = "F"
>>>
```

name

get name of the interface

Returns name of the interface

Return type str

Example

```
>>>
>>> print(int_obj.name)
fc1/1
>>>
```

remove_members(*interfaces*)

Remove Fc members from the port channel

Parameters **interfaces** (*list(Fc)*) – list of Fc interfaces to be removed

Raises **PortChannelNotPresent** – if port channel is not present on switch

Example

```
>>>
>>> pcobj.remove_members([fc1,fc2])
>>>
```

speed

set speed of the interface or get speed of the interface

Getter

Returns speed of the interface

Return type int

Example

```
>>>
>>> print(int_obj.speed)
32000
>>>
```

Setter

Parameters **mode** (*int*) – set speed of the interface

Example

```
>>>
>>> int_obj.speed = 32000
>>>
```

status
set status of the interface or get status of the interface

Getter

Returns status of the interface

Return type str

Example

```
>>>
>>> print(int_obj.status)
trunking
>>>
```

Setter

Parameters **mode** (*str*) – set status of the interface

Values "shutdown", "no shutdown"

Example

```
>>>
>>> int_obj.status = "no shutdown"
>>>
```

trunk
set trunk mode on the interface or get trunk mode on the interface

Getter

Returns trunk mode of the interface

Return type str

Example

```
>>>
>>> print(int_obj.trunk)
on
>>>
```

Setter

Parameters **mode** (*str*) – set trunk mode on the interface

Example

```
>>>
>>> int_obj.trunk = "on"
>>>
```

class mdssdk.interface.Interface.**Counters**(*intobj*)

brief
Get brief counters details of the interface

Returns brief: Returns brief counters details of the interface

Return type dict (name:value)

Example

```
>>>
>>> intcounters = int_obj.counters
>>> print(intcounters.brief)
{'input_rate': 0, 'frames_in': 14970, 'output_rate': 0, 'frames_out
↪': 14831}
>>>
```

clear()

Clear the counters on the interface

Returns None

Example

```
>>>
>>> intcounters = int_obj.counters
>>> intcounters.clear()
>>>
```

congestion_stats

Get congestion stats from the detailed counters of the interface

Returns congestion_stats: congestion stats from the detailed counters of the interface

Return type dict (name:value)

Example

```
>>>
>>> intcounters = int_obj.counters
>>> print(intcounters.congestion_stats)
{'timeout_discards': 0, 'credit_loss': 0, 'bb_scs_resend': 0, 'bb_
↪scr_incr': 0, 'txwait': 0,
'tx_wait_unavbl_1s': 0, 'tx_wait_unavbl_1m': 0, 'tx_wait_unavbl_1hr
↪': 0, 'tx_wait_unavbl_72hr': 0,
'rx_b2b_credit_remain': 1, 'tx_b2b_credit_remain': 0, 'tx_b2b_low_
↪pri_cre': 0, 'rx_b2b_credits': 0, 'tx_b2b_credits': 0}
>>>
```

link_stats

Get link stats from the detailed counters of the interface

Returns link_stats: link stats from the detailed counters of the interface

Return type dict (name:value)

Example

```
>>>
>>> intcounters = int_obj.counters
>>> print(intcounters.link_stats)
{'link_failures': 2, 'sync_loss': 0, 'signal_loss': 0, 'prm_seq_pro_
↪err': 0, 'inv_trans_err': 0,
'inv_crc': 0, 'delim_err': 0, 'frag_frames_rcvd': 0, 'frames_eof_
↪abort': 0, 'unknown_class_frames_rcvd': 0,
```

(continues on next page)

(continued from previous page)

```
'runt_frames': 0, 'jabber_frames': 0, 'too_long': 0, 'too_short': 0,
↪ 'fec_corrected': 0, 'fec_uncorrected': 0,
'rx_link_reset': 0, 'tx_link_reset': 0, 'rx_link_reset_resp': 4,
↪'tx_link_reset_resp': 2, 'rx_off_seq_err': 6,
'tx_off_seq_err': 8, 'rx_non_oper_seq': 3, 'tx_non_oper_seq': 2}
>>>
```

loop_stats

Get loop stats from the detailed counters of the interface

Returns loop_stats: loop stats from the detailed counters of the interface

Return type dict (name:value)

Example

```
>>>
>>> intcounters = int_obj.counters
>>> print(intcounters.loop_stats)
{'rx_f8_lip_seq_err': 0, 'tx_f8_lip_seq_err': 0, 'rx_non_f8_lip_seq_
↪err': 0, 'tx_non_f8_lip_seq_err': 0}
>>>
```

other_stats

Get other stats from the detailed counters of the interface

Returns other_stats: other stats from the detailed counters of the interface

Return type dict (name:value)

Example

```
>>>
>>> intcounters = int_obj.counters
>>> print(intcounters.other_stats)
{'pg_acl_drops': 0, 'pg_fib_start': '1', 'pg_fib_end': '16', 'pg_
↪fib_drops': 0, 'pg_xbar_start': '1',
'pg_xbar_end': '16', 'pg_xbar_drops': 0, 'pg_other_drops': 0}
>>>
```

total_stats

Get total stats from the detailed counters of the interface

Returns total_stats: total stats from the detailed counters of the interface

Return type dict (name:value)

Example

```
>>>
>>> intcounters = int_obj.counters
>>> print(intcounters.total_stats)
{'rx_total_frames': 14970, 'tx_total_frames': 14831, 'rx_total_bytes
↪': 2235488, 'tx_total_bytes': 1733508, 'rx_total_multicast': 0,
'tx_total_multicast': 0, 'rx_total_broadcast': 0, 'tx_total_
↪broadcast': 0, 'rx_total_unicast': 14970, 'tx_total_unicast':␣
↪14831,
'rx_total_discard': 0, 'tx_total_discard': 0, 'rx_total_error': 0,
↪'tx_total_error': 0, 'rx_c_2_frames': 0, 'tx_c_2_frames': 0,
'rx_c_2_bytes': 0, 'tx_c_2_bytes': 0, 'rx_c_2_discards': 0, 'rx_c_2_
↪port_rjt_frames': 0, 'rx_c_3_frames': 14962, 'tx_c_3_frames': ␣
↪14823,
```

(continues on next page)

(continued from previous page)

```
'rx_c_3_bytes': 2235072, 'tx_c_3_bytes': 1733092, 'rx_c_3_discards
↪': 0, 'rx_c_f_frames': 8, 'tx_c_f_frames': 8, 'rx_c_f_bytes': 416,
'tx_c_f_bytes': 416, 'rx_c_f_discards': 0}
>>>
```

2.7 Zone

class mdssdk.zone.**Zone**(*switch*, *name*, *vsan*, *check_npv=True*)

Zone module

Parameters

- **switch** (Switch) – switch object on which zone operations needs to be executed
- **name** (*str*) – zone name with which zone operations needs to be executed
- **vsan** (*int*) – vsan id on which zone operations needs to be executed

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> switch_obj = Switch(ip_address = switch_ip, username = switch_
↪username, password = switch_password)
>>> zoneObj = Zone(switch_obj,"zone_fab_a",1)
>>>
```

active_members

Get active members of the zone i.e zone members part of active zoneset

Returns members: active members of the zone i.e zone members part of active zoneset

Return type list

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zoneObj.active_members)
[{'interface': 'fc1/2'}, {'interface': 'fc1/3'}, {'device-alias':
↪'somename'}, {'pwwn': '11:22:33:44:55:66:77:88'}]
>>>
```

activedb_size

Get active db size of the zone

Returns activedb_size: active db size of the zone, None if no active db

Return type int

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zoneObj.activedb_size)
```

(continues on next page)

(continued from previous page)

```
None
>>>
```

activedb_zone_count
Get active db zone count

Returns activedb_zone_count: active db zone count, None if no active db

Return type int

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zoneObj.activedb_zone_count)
None
>>>
```

activedb_zoneset_count
Get active db zoneset count

Returns activedb_zoneset_count: Returns active db zoneset count, None if no active db

Return type int

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zoneObj.activedb_zoneset_count)
None
>>>
```

activedb_zoneset_name
Get name of the active zoneset

Returns activedb_zoneset_name: name of the active zoneset, else None

Return type str

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zoneObj.activedb_zoneset_name)
None
>>>
```

add_members(*members*)
Add members to the zone

Parameters **members** (*list or dict*) – add members to the zone, there are 2 ways you can add members to the zone (1) a list of members - Fc/Port-channel interface object, device-alias, pwwn or (2) a dict of members - here key will be valid zone member type like “pwwn”,”device-alias”,”interface” etc..

Raises

- **CLIError** – if vsan is not present on the switch

- **InvalidZoneMemberType** – if zone member type is invalid

Example

```
>>>
>>> zoneObj = Zone(switch_obj,"zone_fab_a",1)
>>> zoneObj.create()
>>> int12 = Fc(sw, "fc1/2")
>>> int13 = Fc(sw, "fc1/3")
# add members as a list
>>> zoneObj.add_members([int12, int13, "somename",
↪"11:22:33:44:55:66:77:88"])
>>>
# add members as a dict
>>> memlist = [{'pwwn': '50:08:01:60:08:9f:4d:00'},
... {'pwwn': '50:08:01:60:08:9f:4d:01'},
... {'interface': int13.name},
... {'device-alias': 'hello'}, {'ip-address': '1.1.1.1'},
... {'symbolic-nodename': 'symbnodename'},
... {'fwwn': '11:12:13:14:15:16:17:18'}, {'fcid': '0x123456'},
... {'interface': int12.name},
... {'symbolic-nodename': 'testsymnode'},
... {'fcalias': 'somefcalias'}]
>>> zoneObj.add_members(memlist)
>>>
```

clear_lock()

Clear zone lock if acquired

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> zoneObj.clear_lock()
```

create()

Create zone

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> zoneObj = Zone(switch_obj,"zone_fab_a",1)
>>> zoneObj.create()
>>>
```

default_zone

set default zone or get default zone

Getter

Returns default_zone: default zone status

Return type str

Example

```
>>>
>>> print(zoneObj.default_zone)
```

(continues on next page)

(continued from previous page)

```
deny
>>>
```

Setter

Parameters **default_zone** (*str*) – set default zone value

Values ['permit', 'deny']

Raises

- **CLIError** – if vsan is not present on the switch
- **InvalidDefaultZone** – if def zone value is not ['permit', 'deny']

Example

```
>>>
>>> zoneObj.default_zone = "deny"
>>>
```

delete()

Delete zone

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> zoneObj = Zone(switch_obj,"zone_fab_a",1)
>>> zoneObj.delete()
>>>
```

effectivedb_size

Get effective db size of the zone

Returns effectivedb_size: effective db size of the zone

Return type int

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zoneObj.effectivedb_size)
191
>>>
```

effectivedb_size_percentage

Get effective db size of the zone in percentage terms

Returns effectivedb_size_percentage: Get effective db size of the zone in percentage terms

Return type str

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zoneObj.effectivedb_size_percentage)
0%
>>>
```

fulldb_size
Get full db size of the zone

Returns fulldb_size: full db size of the zone

Return type int

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zoneObj.fulldb_size)
191
>>>
```

fulldb_zone_count
Get full db zone count

Returns fulldb_zone_count: full db zone count

Return type int

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zoneObj.fulldb_zone_count)
1
>>>
```

fulldb_zoneset_count
Get full db zoneset count

Returns fulldb_zoneset_count: full db zoneset count

Return type int

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zoneObj.fulldb_zoneset_count)
0
>>>
```

locked
Check if zone lock is acquired

Returns locked: True if zone lock is acquired else return False

Return type bool

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zoneObj.locked)
False
>>>
```

maxdb_size

Get max db size of the zone

Returns maxdb_size: max db size of the zone

Return type int

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zoneObj.maxdb_size)
4000000
>>>
```

members

Get members of the zone

Returns members: members of the zone

Return type list

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zoneObj.members)
[{'interface': 'fc1/2'}, {'interface': 'fc1/3'}, {'device-alias':
↪'somename'}, {'pwwn': '11:22:33:44:55:66:77:88'}]
>>>
```

mode

set zone mode or get zone mode

Getter

Returns mode: get the current zone mode

Return type str

Example

```
>>>
>>> print(zoneObj.mode)
enhanced
>>>
```

Setter

Parameters **mode** (*str*) – set zone mode

Values ['basic', 'enhanced']

Raises

- **CLIError** – if vsan is not present on the switch

- **InvalidZoneMode** – if zone mode is not ['basic', 'enhanced']

Example

```
>>>
>>> zoneObj.mode = 'enhanced'
>>>
```

name

Get zone name

Returns name: Zone name

Return type str

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> zoneObj = Zone(switch_obj,"zone_fab_a",1)
>>> zoneObj.create()
>>> print(zoneObj.name)
zone_fab_a
>>>
```

remove_members (*members*)

Remove members from the zone

Parameters **members** (*list or dict*) – Remove members from the zone, there are 2 ways you can remove members from the zone (1) a list of members - Fc/Port-channel interface object, device-alias, pwwn or (2) a dict of members - here key will be valid zone member type like "pwwn","device-alias","interface" etc..

Raises

- **CLIError** – if vsan is not present on the switch
- **InvalidZoneMemberType** – if zone member type is invalid

Example

```
>>>
>>> zoneObj = Zone(switch_obj,"zone_fab_a",1)
>>> zoneObj.create()
>>> int12 = Fc(sw, "fc1/2")
>>> int13 = Fc(sw, "fc1/3")
# Remove members as a list
>>> zoneObj.remove_members([int12, int13, "somename",
↪"11:22:33:44:55:66:77:88"])
>>>
# Remove members as a dict
>>> memlist = [{'pwwn': '50:08:01:60:08:9f:4d:00'},
... {'pwwn': '50:08:01:60:08:9f:4d:01'},
... {'interface': int13.name},
... {'device-alias': 'hello'}, {'ip-address': '1.1.1.1'},
... {'symbolic-nodename': 'symbnodename'},
... {'fwwn': '11:12:13:14:15:16:17:18'}, {'fcid': '0x123456'},
... {'interface': int12.name},
... {'symbolic-nodename': 'testsymnode'},
... {'fcalias': 'somefcalias'}]
```

(continues on next page)

(continued from previous page)

```
>>> zoneObj.remove_members(memlist)
>>>
```

smart_zone

set smart zone or get smart zone

Getter

Returns smart_zone : True if smart zone is enabled, False otherwise

Return type bool

Example

```
>>>
>>> print(zoneObj.smart_zone)
True
>>>
```

Setter

Parameters **smart_zone** (*bool*) – enables smart zone if set to True, else disables it

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> zoneObj.smart_zone = True
>>>
```

status

Get the latest status of the zone

Returns status: the latest status of the zone

Return type str

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zoneObj.status)
"Set Smart Zoning Policy complete at 16:03:19 IST Mar 19 2020
>>>
```

vsan

Get vsan object for the zone

Returns vsan: vsan of the zone

Return type *Vsan*

Example

```
>>>
>>> zoneObj = Zone(switch_obj,"zone_fab_a",1)
>>> print(zoneObj.vsan)
<mdslib.vsan.Vsan object at 0x10d105550>
>>> print(zoneObj.vsan.id)
```

(continues on next page)

(continued from previous page)

```
2
>>>
```

2.8 Zoneset

class mdssdk.zoneset.**ZoneSet**(*switch*, *name*, *vsan*)

Zoneset module

Parameters

- **switch** (Switch) – switch object on which zoneset operations needs to be executed
- **name** (*str*) – zoneset name with which zoneset operations needs to be executed
- **vsan** (*int*) – vsan id on which zone operations needs to be executed

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> switch_obj = Switch(ip_address = switch_ip, username = switch_
↪username, password = switch_password)
>>> zonesetObj = ZoneSet(switch_obj,"zoneset_fab_A",1)
>>>
```

activate(*action=True*)

Activate or deactivate a zoneset

Parameters **action** (*bool (deafult: True)*) – activate zoneset if set to True, else deactivate the zoneset

Returns None

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> # Activate the zoneset
>>> zs.activate()
>>> # Deactivate the zoneset
>>> zs.activate(False)
>>>
```

active_members

Get members of the active zoneset if any

Returns members: members active zoneset if any

Return type dict(zone_name: Zone)

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zonesetObj.active_members)
{'zonetemp': <mdslib.zone.Zone object at 0x10dfc3e50>, 'zonetemp_int
↪': <mdslib.zone.Zone object at 0x10dfc3ed0>}
>>>
```

add_members(*members*)
Add members i.e zones to the zoneset

Parameters **members** (*list(Zone)*) – list of Zone members that need to be added to zoneset

Returns None

Raises **CLIError** – If zone is not present in the switch

Example

```
>>>
>>> z1 = Zone(sw,"zonetemp",1)
>>> z2 = Zone(sw,"zonetemp_int",1)
>>> z1.create()
>>> z2.create()
>>> zs = ZoneSet(switch=sw, name="scriptZoneset",vsan=1)
>>> zs.create()
>>> zs.add_members([z1,z2])
>>>
```

create()
Create zoneset

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> zonesetObj = ZoneSet(switch_obj,"zoneset_fab_A",1)
>>> zonesetObj.create()
>>>
```

delete()
Delete zoneset

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> zonesetObj = ZoneSet(switch_obj,"zoneset_fab_A",1)
>>> zonesetObj.delete()
>>>
```

is_active()
Check if the zoneset is active or not

Returns True if zoneset is active, False otherwise

Return type bool

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> zs.is_active()
True
>>>
```

members
Get members of the zoneset

Returns members: members of the zoneset

Return type dict(zone_name: Zone)

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> print(zonesetObj.members)
{'zonetemp': <mdslib.zone.Zone object at 0x10dfc3e50>, 'zonetemp_int
↪': <mdslib.zone.Zone object at 0x10dfc3ed0>}
>>>
```

name
Get zoneset name

Returns name: Zoneset name

Return type str

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> zonesetObj = ZoneSet(switch_obj,"zoneset_fab_A",1)
>>> zonesetObj.create()
>>> print(zonesetObj.name)
zoneset_fab_A
>>>
```

remove_members(*members*)
Remove members i.e zones from the zoneset

Parameters **members** (*list(Zone)*) – list of Zone members that need to be removed from the zoneset

Returns None

Raises **CLIError** – If zone is not present in the switch

Example

```
>>>
>>> zs.remove_members([z1,z2])
>>>
```

vsan
Get vsan object for the zoneset

Returns vsan: vsan of the zoneset

Return type *Vsan*

Raises **CLIError** – if vsan is not present on the switch

Example

```
>>>
>>> zonesetObj = ZoneSet(switch_obj,"zoneset_fab_A",1)
>>> vobj = zonesetObj.vsan
>>> print(vobj)
<mdslib.vsan.Vsan object at 0x10d105550>
>>>
```

2.9 Analytics

class mdssdk.analytics.**Analytics**(*switch*)

Analytics Module

Example

```
>>> switch_obj = Switch(ip_address = switch_ip, username = switch_
↪username, password = switch_password )
>>> ana_hand = switch_obj.analytics
>>> print(ana_hand)
<mdslib.analytics.Analytics object at 0x10ad710d0>
```

clear(*profile*)

clear analytics query

Parameters **profile** (*dict('protocol': value , 'metrics': [values], 'view': value)*) – profile to get the pull query result

Raises **InvalidProfile** – If the profile passed is not correct

Returns switch response to the show query cli and the error if any

Return type tuple: (output, error)

Example

```
>>>
>>> scsi_profile_few = {
... 'protocol': 'scsi',
... 'metrics': ['port', 'total_read_io_count', 'total_write_io_count
↪'],
... 'view': 'port'
... }
>>> ana_hand = switch_obj.analytics
>>> ana_hand.clear(port_scsi_profile)
>>>
```

create_query(*name*, *profile*, *clear=False*, *differential=False*, *interval=30*)

Create analytics query

Parameters

- **name** (*str*) – name of the query to create
- **profile** (*dict('protocol': value , 'metrics': [values], 'view': value)*) – profile for the query

- **clear** (*bool (Default = False)*) – set to True to add clear option to the query else set to False
- **differential** (*bool (Default = False)*) – set to True to add differential option to the query else set to False
- **interval** (*interval (Default = 30)*) – query interval that needs to be set

Raises **InvalidProfile** – If the profile passed is not correct

Returns switch response to the create query cli and the error if any

Return type tuple: (output, error)

Example

```
>>>
>>> port_scsi_profile = {
... 'protocol': 'scsi',
... 'metrics': [],  # default, which is all
... 'view': 'port'
... }
>>> ana_hand = switch_obj.analytics
>>> ana_hand.create_query("port_query",port_scsi_profile)
>>>
```

delete_query(*name*)

Parameters **name** (*str*) – name of the query to delete

Returns switch response to the delete query cli and the error if any

Return type tuple: (output, error)

Example

```
>>>
>>> ana_hand.delete_query(port_scsi_profile)
```

initiators(*module=None*, *protocol=None*)

Get total initiators on the switch or per module

Parameters

- **module** (*int (Default = None)*) – module number for which we need to get total initiators
- **protocol** (*str (Default = None)*) – protocol for which we need to get total initiators if 'scsi' gets scsi initiators, if 'nvme', gets nvme initiators, if None, gets total initiators

Values 'scsi,'nvme',None

Returns total initiators

Return type str

Example

```
>>>
>>> ana_hand = switch_obj.analytics
>>> ana_hand.initiators()
30
```

(continues on next page)

(continued from previous page)

```
>>> ana_hand.initiators(2,'scsi')
10
>>> ana_hand.initiators(2,'nvme')
20
>>>
```

itls(*module=None*)

Get total switch scsi ITLs or total per module scsi ITLs

Parameters **module** (*int (Default = None)*) – module number for which we need to get scsi ITLs, if set to None, get total ITLs of the switch

Returns total ITLs

Return type int

Example

```
>>> ana_hand = switch_obj.analytics
>>> ana_hand.itls()
1248
>>> ana_hand.itls(2)
1000
>>> ana_hand.itls(4)
248
>>>
```

itls_itns(*module=None*)

Get total switch scsi ITLs and nvme ITNs or total per module scsi ITLs and nvme ITNs

Parameters **module** (*int (Default = None)*) – module number for which we need to get scsi ITLs and nvme ITNs, if None gets switch ITLs and ITNs

Returns total scsi ITLs and nvme ITNs

Return type int

Example

```
>>> ana_hand = switch_obj.analytics
>>> ana_hand.itls_itns()
1448
>>> ana_hand.itls_itns(2)
1150
>>> ana_hand.itls_itns(4)
298
>>>
```

itns(*module=None*)

Get total switch nvme ITNs or total per module nvme ITNs

Parameters **module** (*int (Default = None)*) – module number for which we need to get nvme ITNs, if None gets switch nvme ITNs

Returns total ITNs

Return type int

Example

```
>>> ana_hand = switch_obj.analytics
>>> ana_hand.itns()
200
>>> ana_hand.itns(2)
150
>>> ana_hand.itns(4)
50
>>>
```

npu_load (*module*, *protocol=None*)
Get NPU load for a module

Parameters

- **module** (*int*) – module number for which we need to get NPU load
- **protocol** (*str (Default = None)*) – protocol for which NPU load needs to be fetched, options are 'scsi', 'name' or 'None'(both scsi and nvme)

Values 'scsi,'nvme',None

Returns NPU load

Return type str

Example

```
>>>
>>> ana_hand = switch_obj.analytics
>>> ana_hand.npu_load(2)
30%
>>> ana_hand.npu_load(2,'scsi')
10%
>>> ana_hand.npu_load(2,'nvme')
20%
>>>
```

purge (*profile*)
purge analytics query

Parameters **profile** (*dict('protocol': value , 'metrics': [values], 'view': value)*) – profile to get the pull query result

Raises **InvalidProfile** – If the profile passed is not correct

Returns switch response to the show query cli and the error if any

Return type tuple: (output, error)

Example

```
>>>
>>> scsi_profile_few = {
... 'protocol': 'scsi',
... 'metrics': ['port', 'total_read_io_count', 'total_write_io_count
↪'],
... 'view': 'port'
... }
>>> ana_hand = switch_obj.analytics
>>> ana_hand.purge(port_scsi_profile)
>>>
```

show_query (*name=None*, *profile=None*, *clear=False*, *differential=False*)
Get result for installed query or do a pull query

Parameters

- **name** (*str*) – name of the query installed for which result needs to be pulled out
- **profile** (*dict('protocol': value , 'metrics': [values], 'view': value)*) – profile to get the pull query result
- **clear** (*bool (Default = False)*) – set to True to add clear option to the pull query else set to False
- **differential** (*bool (Default = False)*) – set to True to add differential option to the pull query else set to False

Raises **InvalidProfile** – If the profile passed is not correct

Returns switch response to the show query cli and the error if any

Return type tuple: (output, error)

Example

```
>>>
>>> port_scsi_profile = {
... 'protocol': 'scsi',
... 'metrics': [],  # default, which is all
... 'view': 'port'
... }
>>> ana_hand = switch_obj.analytics
>>> ana_hand.create_query("port_query",port_scsi_profile)
>>> out_install = ana_hand.show_query("port_query")
>>> print(out_install)
{'1': {'port': 'fc1/48', 'scsi_target_count': '2', 'scsi_initiator_
↪count': '0', 'io_app_count': '1',
 'logical_port_count': '2', 'scsi_target_app_count': '2',...}
>>> out_pullq = ana_hand.show_query(profile=port_scsi_profile)
>>> print(out_pullq)
{'1': {'port': 'fc1/48', 'scsi_target_count': '2', 'scsi_initiator_
↪count': '0', 'io_app_count': '1',
 'logical_port_count': '2', 'scsi_target_app_count': '2',...}
```

targets (*module=None*, *protocol=None*)
Get total targets on the switch or per module

Parameters

- **module** (*int (Default = None)*) – module number for which we need to get total targets
- **protocol** (*str (Default = None)*) – protocol for which we need to get total targets, options are 'scsi','nvme',None(both scsi and nvme)

Values 'scsi,'nvme',None

Returns total targets

Return type str

Example

```
>>>
>>> ana_hand = switch_obj.analytics
>>> ana_hand.targets()
30
>>> ana_hand.targets(2,'scsi')
10
>>> ana_hand.targets(2,'nvme')
20
>>>
```

2.10 Fdmi

class mdssdk.fdmi.**Fdmi**(*switch*, *hbaid=None*, *vsan=None*)

Fdmi Module

Example

```
>>> switch_obj = Switch(ip_address = switch_ip, username = switch_
↪username, password = switch_password )
>>> fdmi_hand = Fdmi(sw)
>>> print(fdmi_hand)
<mdssdk.fdmi.Fdmi object at 0x103fd5e10>
```

database_detail(*vsan=None*, *hbaid=None*)

Returns all the hba details registered in a dict format

Parameters

- **vsan** (*list*) – vsan list for which details needs to be fetched (optional)
- **hbaid** (*list*) – hbaid list for which details needs to be fetched (optional)

Returns Returns all the hba's discovered

Return type dict(vsan:hba details)

Example

```
>>> allhbas = fdmi.database_detail()
>>> print(allhbas)
{1: [{'current_speed': '32G',
  'driver_ver': '8.07.00.34.Trunk-SCST.18-k',
  'firmware_ver': '8.08.204 (785ad0ult',
  'hardware_ver': 'BK3210407-43 02',
  'hba': '20:05:00:11:0d:fd:4f:00',
  'host_name': 'VirtuaLUN',
  'manufacturer': 'QLogic Corporation',
  'maximum_frame_size': 2112,
  'model': 'QLE2742',
  'model_description': 'Cisco QLE2742 Dual Port 32Gb FC to PCIe␣
↪Gen3 x8 '
                       'Adapter',
  'node_name': '20:05:00:11:0d:fd:4f:00',
  'os_device_name': 'qla2xxx:host14',
  'port': '20:05:00:11:0d:fd:4f:00',
  'rom_ver': '3.39',
  'serial_num': 'RFD1610K18684',
```

(continues on next page)

(continued from previous page)

```
  'supported_fc4_types': ['scsi-fcp', 'NVMe', 'fc-av'],
  'supported_speeds': ['8G', '16G', '32G']}]}
                       '20:07:00:11:0d:60:01:00']}
>>>
```

hbas (*vsan=None*)

Returns all the hba's that are registered

Parameters **vsan** (*list*) – vsan list for which hba list needs to be fetched (optional)

Returns Returns all the hba's that are registered

Return type dict(vsan:hba list)

Example

```
>>> allhbas = fdmi.hbas()
>>> print(allhbas)
{1: ['10:00:00:10:9b:95:41:9c', '20:05:00:11:0d:fd:4f:00'],
 167: ['20:02:00:11:0d:5a:35:00',
       '20:03:00:11:0d:5a:36:00',
       '20:07:00:11:0d:60:01:00']}
>>>
```

CHAPTER 3

Examples

3.1 Switch

```python
from mdssdk.switch import Switch

user = "yourswitchusername"
pw = "yourswitchpassword"
ip_address = "yourswitchip"  # 10.197.155.110
p = 8443

# Set connection_type='https' for NXAPI
# Set connection_type='ssh' for SSH
switch_obj = Switch(
    ip_address=ip_address,
    username=user,
    password=pw,
    connection_type="https",
    port=p,
    timeout=30,
    verify_ssl=False,
)

# Displaying switch name, version, image
print("Name: " + switch_obj.name)
print("Version: " + switch_obj.version)
print("Kickstart Image: " + switch_obj.kickstart_image)
print("System Image: " + switch_obj.system_image)

# Changing name of switch
switch_obj.name = "switch_test"
print("Changed Name: " + switch_obj.name)

# Enabling feature analytics
switch_obj.feature("analytics", True)
```

(continues on next page)

(continued from previous page)

```
print("Analytics feature : " + str(switch_obj.feature("analytics")))
```

Download: ExampleScript

3.2 Module

```
from mdssdk.switch import Switch

user = "yourswitchusername"
pw = "yourswitchpassword"
ip_address = "yourswitchip"  # 10.197.155.110
p = 8443

# Set connection_type='https' for NXAPI
# Set connection_type='ssh' for SSH
switch_obj = Switch(
    ip_address=ip_address,
    username=user,
    password=pw,
    connection_type="https",
    port=p,
    timeout=30,
    verify_ssl=False,
)

# Print the information of all switch modules
mod_handler = switch_obj.modules  # dict of module objects
print(mod_handler)
for m in mod_handler.values():
    print("Model : " + m.model)
    print("Module Number : " + str(m.module_number))
    print("Ports : " + str(m.ports))
    print("Status : " + m.status)
    print("Type : " + m.type)
```

Download: ExampleScript

3.3 Vsan

```
from mdssdk.switch import Switch
from mdssdk.vsan import Vsan

user = "yourswitchusername"
pw = "yourswitchpassword"
ip_address = "yourswitchip"  # 10.197.155.110
p = 8443

# Set connection_type='https' for NXAPI
# Set connection_type='ssh' for SSH
sw = Switch(
    ip_address=ip_address,
    username=user,
```

(continues on next page)

(continued from previous page)

```
    password=pw,
    connection_type="https",
    port=p,
    timeout=30,
    verify_ssl=False,
)

# Example for creating and deleting 10 vsan objects from id 10 to 19
vsan = []
for i in range(10, 20):
    vsan.append(Vsan(switch=sw, id=i))
print("Vsan ID\tName\tState")
for v in vsan:
    v.create()  # creates vsan on switch
    print(str(v.id) + "\t\t" + v.name + "\t" + v.state)  # print id,name,state
    v.delete()  # deletes vsan
```

Download: `ExampleScript`

3.4 DeviceAlias

```
from mdssdk.devicealias import DeviceAlias
from mdssdk.switch import Switch

user = "yourswitchusername"
pw = "yourswitchpassword"
ip_address = "yourswitchip"  # 10.197.155.110
p = 8443

# Set connection_type='https' for NXAPI
# Set connection_type='ssh' for SSH
sw = Switch(
    ip_address=ip_address,
    username=user,
    password=pw,
    connection_type="https",
    port=p,
    timeout=30,
    verify_ssl=False,
)

# Instantiating DeviceAlias object
d = DeviceAlias(sw)

# Display the database, mode, distribute, locked
print("Device Alias Database")
print(d.database)
print("Mode : " + d.mode)
print("Distribute : " + str(d.distribute))
print("Locked : " + str(d.locked))

old = d.database
d.clear_database()

```

(continues on next page)

(continued from previous page)

```
# Adding new device alias
new = {"device1": "21:00:00:0e:1e:30:34:a5", "device2": "21:00:00:0e:1e:30:3c:c5"}
d.create(new)

prnt("Clearing database\nDatabase after adding new entry")
print(d.database)

# Renaming the device alias
d.rename("device1", "device_new_name")

print("Database after renaming device alias device1 as device_new_name")
print(d.database)

# Deleting device alias
d.delete("device_new_name")
d.delete("device2")

# Recreating original database
d.create(old)
```

Download: `ExampleScript`

3.5 Zone

```
from mdssdk.switch import Switch
from mdssdk.vsan import Vsan
from mdssdk.zone import Zone

user = "yourswitchusername"
pw = "yourswitchpassword"
ip_address = "yourswitchip"  # 10.197.155.110
p = 8443

# Set connection_type='https' for NXAPI
# Set connection_type='ssh' for SSH
sw = Switch(
    ip_address=ip_address,
    username=user,
    password=pw,
    connection_type="https",
    port=p,
    timeout=30,
    verify_ssl=False,
)

# Instantiating Vsan object with id 2
v = Vsan(sw, 2)

# Creating vsan
v.create()

# Instantiate zone object
z = Zone(sw, "zone1", v.id)

```

(continues on next page)

(continued from previous page)

```
# Create new zone
z.create()

memlist = [
    {"pwwn": "50:08:01:60:08:9f:4d:00"},
    {"ip-address": "1.1.1.1"},
    {"symbolic-nodename": "symbnodename"},
    {"fwwn": "11:12:13:14:15:16:17:18"},
    {"fcid": "0x123456"},
]

# Adding members to zone
z.add_members(memlist)

# Display zone name, vsan id, members
print("Zone name: " + z.name)
print("Vsan id: " + str(z.vsan.id))
print("Zone members: " + str(z.members))

# Removing members from zone
z.remove_members(memlist)

# Deleting zone
z.delete()

# Deleting vsan
v.delete()
```

Download: `ExampleScript`

3.6 Zoneset

```
from mdssdk.devicealias import DeviceAlias
from mdssdk.fc import Fc
from mdssdk.portchannel import PortChannel
from mdssdk.switch import Switch
from mdssdk.vsan import Vsan
from mdssdk.zone import Zone
from mdssdk.zoneset import ZoneSet

# Switch credentials
user = "yourswitchusername"
pw = "yourswitchpassword"
ip_address = "yourswitchip"  # 10.197.155.110
p = 8443

# Creating switch object
sw = Switch(
    ip_address=ip_address,
    username=user,
    password=pw,
    connection_type="https",
    port=p,
    timeout=30,
```

(continues on next page)

(continued from previous page)

```
    verify_ssl=False,
)

# Instantiating Vsan object with id 2
v = Vsan(sw, 2)

# Creating vsan
v.create()

# Creating Fc object for interface fc1/3
int13 = Fc(sw, "fc1/3")

# Instantiating PortChannel object 1
pc1 = PortChannel(sw, 1)

# Creating port channel
pc1.create()

# Adding interfaces to vsan 2
v.add_interfaces([int13, pc1])

# Instantiating DeviceAlias object
d = DeviceAlias(sw)
new = {"da1": "60:66:61:01:0e:00:01:ff"}

# Adding new device alias
d.create(new)

# Instantiate zone object
z = Zone(sw, "zone1", v.id)

# Create new zone
z.create()

# Configuring fcalias
sw.config("fcalias name somefcalias vsan " + str(v.id))

memlist = [
    {"pwwn": "50:08:01:60:08:9f:4d:00"},
    {"pwwn": "50:08:01:60:08:9f:4d:01"},
    {"interface": int13.name},
    {"device-alias": "da1"},
    {"ip-address": "1.1.1.1"},
    {"symbolic-nodename": "symbnodename"},
    {"fwwn": "11:12:13:14:15:16:17:18"},
    {"fcid": "0x123456"},
    {"interface": pc1.name},
    {"symbolic-nodename": "testsymnode"},
    {"fcalias": "somefcalias"},
]

# Adding members to zone
z.add_members(memlist)

# Instantiating ZoneSet object
zoneset = ZoneSet(sw, "zoneset1", v.id)

```

(continues on next page)

(continued from previous page)

```
# Creating zoneset
zoneset.create()

# Add members to zoneset
zoneset.add_members([z])

# Activating zoneset
zoneset.activate(True)

# Display zoneset information
print("Zoneset name: " + zoneset.name)
print("Vsan id: " + str(zoneset.vsan.id))
print("Zoneset members: " + str(zoneset.members))
print("Activation: " + zonese.is_active())

# Removing members from zoneset
zoneset.remove_members([z])

# Deleting zoneset
zoneset.delete()

# Removing members from zone
z.remove_members(memlist)

# Deleting zone
z.delete()

# Deleting vsan
v.delete()

# Deleting device alias
d.delete("da1")

# Deleting port channel
pc1.delete()
```

Download: `ExampleScript`

3.7 Multiprocessing

```
# An example to show how we can do multiprocessing to execute some logic on multiple␣
↪switches parallelly

from mdssdk.switch import Switch
from concurrent.futures import wait
from concurrent.futures.thread import ThreadPoolExecutor
import multiprocessing

user = "your_switch_username"
pw = "your_switch_password"
iplist = ["ip1", "ip2"]
p = 8443

myData = {}
```

(continues on next page)

(continued from previous page)

```
14
15
16  def runAnySwitchLogic(ip, user, pw, port):
17      my_switch = Switch(ip, user, pw, "https", port=port, verify_ssl=False)
18      status = isCFSoIpEnabled(my_switch)
19      myData[my_switch.name] = status
20
21
22  def isCFSoIpEnabled(sw):
23      cmd = "show cfs status"
24      # ensures that the output is in cli output format
25      out = sw.show(cmd, raw_text=True)
26      if "Distribution over IP : Disabled" in out:
27          return "Disabled"
28      return "Enabled"
29
30
31  m = multiprocessing.Manager()
32  allfutures = []
33  executor = ThreadPoolExecutor(len(iplist))
34  IP_list = []
35  CFS_list = []
36
37  for i in range(0, len(iplist)):
38      ip = iplist[i]
39      fut = executor.submit(runAnySwitchLogic, ip, user, pw, p)
40      allfutures.append(fut)
41  wait(allfutures)
42
43  for swname, cfsstatus in myData.items():
44      print('CFS Distribution over IP on switch', swname, 'is', cfsstatus)
```

Download: ExampleScript

CHAPTER 4

Credits

4.1 Contributors

- Suhas Bharadwaj<subharad@cisco.com>

CHAPTER 5

History

5.1 v1.4.0 (2022-1-27)

- Support for installation via pip install
- Fix analytics setter api
- added setup.cfg for pypi upload

5.2 v1.3.0 (2021-8-23)

- Limited Support for N9K and FI
- Some bug fixes and enhancements

5.3 v1.2.0 (2021-2-17)

- Support for 8.5(1) release
- Some bug fixes and improvements

5.4 v1.1.0 (2020-08-21)

- Support for 8.4(2b) release
- Many bug fixes and improvements

5.5 v1.0.1 (2020-05-11)

- Python SDK/API library for Cisco MDS switches
- PEP8 Compliance
- Supported modules are switch,devicealias,vsan,zone,zoneset and many more
- Please check the documentation for more details.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index

T

V

Z