Pytanga

Renato Almeida de Oliveira

Jan 03, 2021

USING PYTANGA

Pytanga is a Python library that aims to simplify YANG payload creation, its architecture is based on the Composite and Visitor design patterns.

Similar to YANG models where a container can have leaves and other containers, building a part-whole architecture. Pitanga modules define a component with attributes (leaves), and children that represent the inner containers of the module.

For the data output, Pytanga implements a Visitor Pattern that is injected in each component and build the desired output, currently implemented only for NETCONF.

With that architecture, it is possible to define the YANG models' logic and syntax tests decoupled of the payload generation.

CHAPTER

ONE

QUICK START

1.1 Installing

```
pip install pytanga
```

1.2 Basic Usage

Pytanga uses a Composite pattern to abstract YANG models, so each component has an add method which is we ed to compose the payload.

Therefore, to build the desired payload it will be necessary to instantiate all modules equivalent to the YANG models and use the add method building the necessary hierarchy, as the example bellow:

```
from pytanga.components import configComponent
from pytanga.components.OpenConfig.routing import networkInstancesComponent
from pytanga.components.OpenConfig.routing import networkInstanceComponent
from pytanga.components.OpenConfig.routing import protocolsComponent
from pytanga.components.OpenConfig.routing import protocolComponent
from pytanga.components.OpenConfig.routing.static import staticroutesComponent
from pytanga.components.OpenConfig.routing.static import staticComponent
from pytanga.components.OpenConfig.routing.static import nexthopsComponent
from pytanga.components.OpenConfig.routing.static import nexthopComponent
from pytanga.components.OpenConfig.routing.static import interfacerefComponent
from pytanga.visitors import NETCONFVisitor
from xml.dom.minidom import parseString

config = configComponent()
netInsts = networkInstancesComponent()
netinst = networkInstanceComponent()
protos = protocolsComponent()
proto = protocolComponent(identifier ='STATIC' , name='DEFAULT')
staticroutes = staticroutesComponent()
static = staticComponent(prefix= '172.30.0.0/24')
nexthops = nexthopsComponent()
nexthop = nexthopComponent(index='NETCONF' , next_hop='192.168.0.4')

nexthops.add(nexthop)
static.add(nexthops)
staticroutes.add(static)
proto.add(staticroutes)
protos.add(proto)
```

(continues on next page)

(continued from previous page)

```
netinst.add(protos)
netInsts.add(netinst)
config.add(netInsts)

serializer = NETCONFVisitor()
output = static.parse(serializer)
xml_string = serializer.print(output)
print(parseString(xml_string).toprettyxml())
```

Resulting in the following output

```
<static>
    <prefix>172.30.0.0/24</prefix>
    <config>
        <prefix>172.30.0.0/24</prefix>
    </config>
    <next-hops>
        <next-hop>
            <index>NETCONF</index>
            <config>
                <index>NETCONF</index>
                <next-hop>192.168.0.4</next-hop>
            </config>
        </next-hop>
    </next-hops>
</static>
```

1.3 Configuring BGP

For configure BGP use the *ConfigureBGP* Helper

```
from pytanga.components import configComponent
from pytanga.components.Cisco.xe import nativeComponent
from pytanga.components.Cisco.xe import routerComponent
from pytanga.helpers.Cisco.xe import ConfigureBGP
from pytanga.visitors import NETCONFVisitor
from xml.dom.minidom import parseString

BGPHelper = ConfigureBGP(asn=100 , router_id='10.0.0.2')
BGPHelper.addAfi_Safi(afi_name ='ipv4' , safi_name='unicast')
BGPHelper.addNeighbor(address= '10.0.0.1'  , remote_as='100')
BGPHelper.addNeighbor(address= '10.0.0.3'  , remote_as='100')
BGPHelper.addAfiSafiNeighbor(afi_safi='ipv4-unicast' , address= '10.0.0.1')
BGPHelper.addAfiSafiNeighbor(afi_safi='ipv4-unicast' , address= '10.0.0.3')
BGPHelper.addAfiSafiNeighborRouteMap(afi_safi='ipv4-unicast' , nei_address= '10.0.0.3
↪' , inout='in' , name='Teste')
BGPHelper.addAfiSafiNetwork(afi_safi='ipv4-unicast' , network="10.0.0.0" , mask='255.
↪255.255.255')
BGP = BGPHelper.getBGP()

router = routerComponent()
XENative = nativeComponent()
config = configComponent()
```

(continues on next page)

(continued from previous page)

```
router.add(BGP)
XENative.add(router)
config.add(XENative)

serializer = NETCONFVisitor()
output = config.parse(serializer)
xml_string = serializer.print(output)
print(parseString(xml_string).toprettyxml())
```

Resulting in the following output

```
<config>
    <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <router>
            <bgp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-bgp">
                <id>100</id>
                <bgp>
                    <router-id>
                        <ip-id>10.0.0.1</ip-id>
                    </router-id>
                </bgp>
                <address-family>
                    <no-vrf>
                        <ipv4>
                            <af-name>unicast</af-name>
                            <ipv4-unicast>
                                <neighbor>
                                    <id>10.0.0.2</id>
                                </neighbor>
                                <neighbor>
                                    <id>10.0.0.3</id>
                                    <route-map>
                                        <inout>in</inout>
                                        <route-map-name>Teste</route-map-name>
                                    </route-map>
                                </neighbor>
                                <network>
                                    <with-mask>
                                        <number>10.0.0.0</number>
                                        <mask>255.255.255.255</mask>
                                    </with-mask>
                                </network>
                            </ipv4-unicast>
                        </ipv4>
                    </no-vrf>
                </address-family>
                <neighbor>
                    <id>10.0.0.2</id>
                    <remote-as>100</remote-as>
                </neighbor>
                <neighbor>
                    <id>10.0.0.3</id>
                    <remote-as>100</remote-as>
                </neighbor>
            </bgp>
        </router>
    </native>
```

(continues on next page)

(continued from previous page)

```
</config>
```

IOS-XE 16.9.1 resulting configuration:

```
router bgp 100
 bgp router-id 10.0.0.2
 bgp log-neighbor-changes
 neighbor 10.0.0.1 remote-as 100
 neighbor 10.0.0.3 remote-as 100
 !
 address-family ipv4
  network 10.0.0.0 mask 255.255.255.255
  neighbor 10.0.0.1 activate
  neighbor 10.0.0.3 activate
  neighbor 10.0.0.3 route-map Teste in
 exit-address-family
```

1.4 Configuring Prefix-List

For configure prefix list use the *ConfigurePrefixList* Helper

```
from pytanga.components import configComponent
from pytanga.components.Cisco.xe import nativeComponent
from pytanga.components.Cisco.xe.ip import ipComponent
from pytanga.helpers.Cisco.xe import ConfigurePrefixList
from pytanga.visitors import NETCONFVisitor
from xml.dom.minidom import parseString

config = configComponent()
native = nativeComponent()
ip = ipComponent()
PrefixListHelper = ConfigurePrefixList(name="TEST-AS")
PrefixListHelper.addPrefix(action="permit" , network="10.0.40.0/24")
PrefixListHelper.addPrefix(action="permit" , network="10.0.50.0/24")

ip.add(PrefixListHelper.getPrefixList())
native.add(ip)
config.add(native)

serializer = NETCONFVisitor()
output = config.parse(serializer)
xml_string = serializer.print(output)
print(parseString(xml_string).toprettyxml())
```

Resulting in the following output

```
<config>
    <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <ip>
            <prefix-list>
                <prefixes>
                    <name>TEST-AS</name>
                    <seq>
                        <no>5</no>
```

(continues on next page)

(continued from previous page)

```
                        <permit>
                            <ip>10.0.40.0/24</ip>
                        </permit>
                    </seq>
                    <seq>
                        <no>10</no>
                        <permit>
                            <ip>10.0.50.0/24</ip>
                        </permit>
                    </seq>
                </prefixes>
            </prefix-list>
        </ip>
    </native>
</config>
```

IOS-XE 16.9.1 resulting configuration:

```
ip prefix-list TEST-AS seq 5 permit 10.0.40.0/24
ip prefix-list TEST-AS seq 10 permit 10.0.50.0/24
```

1.5 Configuring IP Interface with OpenConfig

For configure an IP Interface use the *CreateIPInterface* Helper

```
from pytanga.components import configComponent
from pytanga.helpers.OpenConfig.interfaces import createIPInterface
from pytanga.components.OpenConfig.interfaces import interfacesComponent
from pytanga.visitors import NETCONFVisitor
from xml.dom.minidom import parseString

interfaces = interfacesComponent()
interface = createIPInterface(name="GigabitEthernet2",
                  if_type='ethernet',
                  ip_version=4,
                  address='10.0.0.5',
                  prefix_length=30,
                  if_mtu= 1650,
                  if_description='Test Configuration',
                  enabled=True)

interfaces.add(interface)
config = configComponent()
config.add(interfaces)
serializer = NETCONFVisitor()
output = config.parse(serializer)
xml_string = serializer.print(output)
print(parseString(xml_string).toprettyxml())
```

Resulting in the following output

```
<config>
    <interfaces xmlns="http://openconfig.net/yang/interfaces" xmlns:oc-ip="http://
↪openconfig.net/yang/interfaces/ip">
```

(continues on next page)

(continued from previous page)

```
        <interface>
            <name>GigabitEthernet2</name>
            <config>
                <description>Test Configuration</description>
                <mtu>1650</mtu>
                <enabled>true</enabled>
                <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
↪ianaift:ethernetCsmacd</type>
            </config>
            <subinterfaces>
                <subinterface>
                    <index>0</index>
                    <oc-ip:ipv4>
                        <oc-ip:addresses>
                            <oc-ip:address>
                                <oc-ip:ip>10.0.0.5</oc-ip:ip>
                                <oc-ip:config>
                                    <oc-ip:ip>10.0.0.5</oc-ip:ip>
                                    <oc-ip:prefix-length>30</oc-ip:prefix-length>
                                </oc-ip:config>
                            </oc-ip:address>
                        </oc-ip:addresses>
                    </oc-ip:ipv4>
                </subinterface>
            </subinterfaces>
        </interface>
    </interfaces>
</config>
```

IOS-XE 16.9.1 resulting configuration:

```
interface GigabitEthernet2
 description Test Configuration
 mtu 1650
 ip address 10.0.0.5 255.255.255.252
end
```

CHAPTER

TWO

COMPONENT CREATION

For create a custom component it is required to create a new implementation of the *AbstractComponent*

In the __init__ method the class should receive all the desired attributes for the YANG module, and pass then to the setAttributes method that will perform the data validation and construction of the modules attribute schema.

Set the attribute self.tag to the model desired tag, and if required set self._xmlns to the XMLNS with the following syntax:

```
self._xmlns = {
    'xmlns' : "http://cisco.com/ns/yang/Cisco-IOS-XE-native" ,
}
```

Warning: The NETCONFVisitor only supports three levels of attributes nesting. If you need more than four levels consider building a new module.

2.1 Implemantation Example

```
from pytanga.components import AbstractComponent


class testComponent(AbstractComponent):

    def __init__(self, attribute1 , attribute2):
        self._xmlns = {}
        self.attributes = self.setAttributes(attribute1 , attribute2)
        self.parent_xmlns = {}
        self._children: List[AbstractComponent] = []
        self.childrenData = []
        self.tag = 'test'

    @property
    def xmlns(self):
        return self._xmlns

    @xmlns.setter
    def xmlns(self, xmlns):
        self._xmlns = xmlns

    def setAttributes(self, attribute1 , attribute2):
        attributes = {}
```

(continues on next page)

(continued from previous page)

```
        attributes['l1_var1'] = attribute1
        attributes['l1_var2'] = 'var2'
        attributes['l1_var3'] = None
        attributes['l1_var4'] = {
            'keys': {
                'attr_key' : attribute2
            },
            'value' : 'var4'
        }
        attributes['level2'] = {
            'l2_var1': 'l2_var1',
            'l2_var2': 'l2_var2',
            'l2_var3': None,
            'l2_var4': {
                'keys': {
                    'testkey': "value"
                },
                'value' : 'var4'
            }
        }
        attributes['level3'] = {
            'level3': {
                'l3_var1': 'l3_var1',
                'l3_var2' : None,
                'l3_var3' : {
                    'keys' : {
                        'l3_key' : 'key',
                    },
                    'value' : 'var3'
                }
            }
        }

        return attributes

    def add(self, component) -> None:
        self._children.append(component)

    def remove(self, component) -> None:
        self._children.remove(component)

    def is_composite(self) -> bool:
        return False

    def getXMLNS(self):
        childrenData = []
        for child in self._children:
            self.parent_xmlns.update(child.getXMLNS())
        return self.parent_xmlns

    def parse(self, serializer):
        self.childrenData = []
        self.getXMLNS()
        for child in self._children:
            self.childrenData.append(child.parse(serializer))
        return serializer.parse(self)
```

2.2 Using the new Module

```
from newmodule import testComponent
from pytanga.visitors import NETCONFVisitor
from xml.dom.minidom import parseString

module = testComponent("Value1" , "Value2")
serializer = NETCONFVisitor()
output = module.parse(serializer)
xml_string = serializer.print(output)
print(parseString(xml_string).toprettyxml())
```

Resulting in the following output

```
<test>
    <l1_var1>Value1</l1_var1>
    <l1_var2>var2</l1_var2>
    <l1_var3/>
    <l1_var4 attr_key="Value2">var4</l1_var4>
    <level2>
        <l2_var1>l2_var1</l2_var1>
        <l2_var2>l2_var2</l2_var2>
        <l2_var3/>
        <l2_var4 testkey="value">var4</l2_var4>
    </level2>
    <level3>
        <level3>
            <l3_var1>l3_var1</l3_var1>
            <l3_var2/>
            <l3_var3 l3_key="key">var3</l3_var3>
        </level3>
    </level3>
</test>
```

CHAPTER

THREE

COMPONENTS

3.1 Cisco

3.1.1 XE

native

class pytanga.components.Cisco.xe.native.**nativeComponent**
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

property xmlns

router

class pytanga.components.Cisco.xe.router.**routerComponent**(*operation=None*)
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes()

property xmlns

IP

ip

class pytanga.components.Cisco.xe.ip.ip.**ipComponent**
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes()

property xmlns

prefix

class pytanga.components.Cisco.xe.ip.prefix.**prefixComponent**(*seq*, *action*, *network*, *ge=None*, *le=None*)
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*seq*, *action*, *network*, *ge*, *le*)

property xmlns

exception pytanga.components.Cisco.xe.ip.prefix.**prefixSyntaxError**
Bases: Exception

prefixlist

class pytanga.components.Cisco.xe.ip.prefixlist.**prefixlistComponent**(*name*, *description=None*, *operation=None*)

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*name*, *description*)

property xmlns

prefixlists

class pytanga.components.Cisco.xe.ip.prefixlists.**prefixeslistsComponent**(*sequence_number=None*, *operation=None*)

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*sequence_number*)

property xmlns

routemap

class pytanga.components.Cisco.xe.ip.routemap.**routemapComponent**(*name*)
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*name*)

property xmlns

routemapentry

exception pytanga.components.Cisco.xe.ip.routemapentry.**routemapSyntaxError**
Bases: Exception

class pytanga.components.Cisco.xe.ip.routemapentry.**routemapentryComponent**(*operation*, *description=None*, *seq_no=None*, *ordering_seq=None*)
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*operation*, *description*, *seq_no*, *ordering_seq*)

property xmlns

BGP

addressFamilyIPv4Unicast

class pytanga.components.Cisco.xe.bgp.addressFamilyIPv4Unicast.**addressFamilyIPv4UnicastComp**

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*auto_summary*, *originate_default*, *default_metric*, *synchronization*, *segment_routing_mpls*)

property xmlns

addressFamilyType

class pytanga.components.Cisco.xe.bgp.addressFamilyType.**addressFamilyTypeComponent**(*afi_name*, *safi_name*)

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*safi_name*)

property xmlns

exception pytanga.components.Cisco.xe.bgp.addressFamilyType.**addressFamilyTypeSyntaxError**
Bases: Exception

addressFamilyVRF

class pytanga.components.Cisco.xe.bgp.addressFamilyVRF.**addressFamilyVRFComponent**(*with_vrf*)
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes()

property xmlns

addressfamilies

class pytanga.components.Cisco.xe.bgp.addressfamilies.**addressFamiliesComponent**
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes()

property xmlns

bgp

class pytanga.components.Cisco.xe.bgp.bgp.**bgpComponent**(*asn*, *aigp_rib_metric=None*, *always_compare_med=None*, *cluster_id=None*, *deterministic_med=None*, *enforce_first_as=None*, *enhanced_error=None*, *fast_external_fallover=None*, *log_neighbor_changes=None*, *maxas_limit=None*, *maxcommunity_limit=None*, *route_map_cache=None*, *update_delay=None*, *router_id=None*)

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*asn*)

property xmlns

class pytanga.components.Cisco.xe.bgp.bgp.**bgpConfigComponent**(*aigp_rib_metric=None*, *always_compare_med=None*, *cluster_id=None*, *deterministic_med=None*, *enforce_first_as=None*, *enhanced_error=None*, *fast_external_fallover=None*, *log_neighbor_changes=None*, *maxas_limit=None*, *maxcommunity_limit=None*, *route_map_cache=None*, *update_delay=None*, *router_id=None*, *advertise_best_external=None*, *dmzlink_bw=None*, *suppress_inactive=None*, *soft_reconfig_backup=None*, *scan_time=None*)

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*aigp_rib_metric*, *always_compare_med*, *cluster_id*, *deterministic_med*, *enforce_first_as*, *enhanced_error*, *fast_external_fallover*, *log_neighbor_changes*, *maxas_limit*, *maxcommunity_limit*, *route_map_cache*, *update_delay*, *router_id*, *advertise_best_external*, *dmzlink_bw*, *suppress_inactive*, *soft_reconfig_backup*, *scan_time*)

property xmlns

neighbor

class pytanga.components.Cisco.xe.bgp.neighbor.**neighborComponent**(*address=None*, *name=None*, *remote_as=None*, *cluster_id=None*, *description=None*, *disable_connected_check=None*, *ebgp_multihop=None*, *password=None*, *peer_group=None*, *shutdown=None*, *keepalive_interval=None*, *holdtime=None*, *minimum_neighbor_hold=None*, *ttl_security=None*, *update_source=None*, *version=None*, *activate=None*, *advertisement_interval=None*, *allow_policy=None*, *allowas_in=None*, *default_originate=None*, *default_originate_route_map=None*, *dmzlink_bw=None*, *maximum_prefix_n=None*, *maximum_prefix_threshold=None*, *maximum_prefix_restart=None*, *maximum_prefix_warning=None*, *next_hop_self=None*, *next_hop_self_all=None*, *next_hop_unchanged=None*, *route_reflector_client=None*, *send_community=None*, *send_label=None*, *soft_reconfiguration=None*, *weight=None*)

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*address*, *name*, *remote_as*, *cluster_id*, *description*, *disable_connected_check*, *ebgp_multihop*, *password*, *peer_group*, *shutdown*, *keepalive_interval*, *holdtime*, *minimum_neighbor_hold*, *ttl_security*, *update_source*, *version*, *activate*, *advertisement_interval*, *allow_policy*, *allowas_in*, *default_originate*, *default_originate_route_map*, *dmzlink_bw*, *maximum_prefix_n*, *maximum_prefix_threshold*, *maximum_prefix_restart*, *maximum_prefix_warning*, *next_hop_self*, *next_hop_self_all*, *next_hop_unchanged*, *route_reflector_client*, *send_community*, *send_label*, *soft_reconfiguration*, *weight*)

property xmlns

exception pytanga.components.Cisco.xe.bgp.neighbor.**neighborSyntaxError**
Bases: Exception

neighborAdvertiseMap

class pytanga.components.Cisco.xe.bgp.neighborAdvertiseMap.**neighborAdvertiseMapComponent**(*na*
ex
isi
na

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*name*, *exist_map*, *non_exist_map*)

property xmlns

neighborAdvertiseMaps

class pytanga.components.Cisco.xe.bgp.neighborAdvertiseMaps.**neighborAdvertiseMapsComponent**
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes()

property xmlns

neighborDistributeList

class pytanga.components.Cisco.xe.bgp.neighborDistributeList.**neighborDistributeListCompone**

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*inout*, *name*)

property xmlns

neighborPrefixList

class pytanga.components.Cisco.xe.bgp.neighborPrefixList.**neighborPrefixListComponent**(*inout*, *name*)
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*inout*, *name*)

property xmlns

neighborRouteMap

class pytanga.components.Cisco.xe.bgp.neighborRouteMap.**neighborRouteMapComponent**(*inout*, *name*)

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*inout*, *name*)

property xmlns

network

class pytanga.components.Cisco.xe.bgp.network.**networkComponent**(*network*, *mask=None*, *route_map=None*, *backdoor=None*)

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*network*, *mask*, *route_map*, *backdoor*)

property xmlns

peergroup

class pytanga.components.Cisco.xe.bgp.peergroup.**peerGroupComponent**(*name*, *remote_as=None*, *cluster_id=None*, *description=None*, *disable_connected_check=None*, *ebgp_multihop=None*, *password=None*, *shutdown=None*, *keepalive_interval=None*, *holdtime=None*, *minimum_neighbor_hold=None*, *ttl_security=None*, *update_source=None*, *version=None*)

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes()

property xmlns

OSPF

Area

class pytanga.components.Cisco.xe.ospf.area.**areaComponent**(*area_id*, *default_cost=None*)

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*area_id*, *default_cost*)

property xmlns

Network

class pytanga.components.Cisco.xe.ospf.network.**networkComponent**(*network*, *wildcard_mask*, *area=None*)
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*ip*, *wildcard_mask*, *area*)

property xmlns

ospf

class pytanga.components.Cisco.xe.ospf.ospf.**ospfComponent**(*process_id*, *vrf=None*, *router_id=None*, *nsr=None*, *maximum_paths=None*, *domain_tag=None*, *ispf=None*, *prefix_suppression=None*, *priority=None*, *shutdown=None*, *cost=None*, *flood_reduction=None*, *hello_interval=None*, *mtu_ignore=None*, *resync_timeout=None*, *retransmit_interval=None*, *transmit_delay=None*)
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*process_id*, *vrf*, *router_id*, *nsr*, *maximum_paths*, *domain_tag*, *ispf*, *prefix_suppression*, *priority*, *shutdown*, *cost*, *flood_reduction*, *hello_interval*, *mtu_ignore*, *resync_timeout*, *retransmit_interval*, *transmit_delay*)

property xmlns

3.1.2 XR

Module contents

3.2 OpenConfig

3.2.1 Interfaces

Ethernet

class pytanga.components.OpenConfig.interfaces.ethernet.**ethernetComponent**(*mac_address=None*, *auto_negotiate=None*, *duplex_mode=None*, *port_speed=None*, *enable_flow_control=None*)

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*mac_address*, *auto_negotiate*, *duplex_mode*, *port_speed*, *enable_flow_control*)

property xmlns

Interface

class pytanga.components.OpenConfig.interfaces.interface.**interfaceComponent**(*name*, *if_type*, *if_description=None*, *if_mtu=None*, *enabled=None*)

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

IETF_INTERFACE_TYPES = {'ethernet': 'ianaift:ethernetCsmacd', 'loopback': 'ianaift:s

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*name*, *if_type*, *if_description*, *if_mtu*, *enabled*)

property xmlns

Interfaces

class pytanga.components.OpenConfig.interfaces.interfaces.**interfacesComponent**
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

property xmlns

oc_ip

class pytanga.components.OpenConfig.interfaces.oc_ip.**oc_ipComponent**(*version*)
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

property parent

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

property xmlns

oc_ipAddress

class pytanga.components.OpenConfig.interfaces.oc_ipAddress.**oc_ipAddressComponent**(*address*, *prefix_length*)

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

property parent

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*address*, *prefix_length*)

property xmlns

oc_ipAddresses

class pytanga.components.OpenConfig.interfaces.oc_ipAddresses.**oc_ipAddressesComponent**

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

property parent

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

property xmlns

subinterface

class pytanga.components.OpenConfig.interfaces.subinterface.**subinterfaceComponent**(*index*, *description=None*)

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*index*, *description*)

property xmlns

subinterfaces

class pytanga.components.OpenConfig.interfaces.subinterfaces.**subinterfacesComponent**

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

property xmlns

switchedVlan

class pytanga.components.OpenConfig.interfaces.switchedVlan.**switchedVlanComponent**(*interface_mo*, *native_vlan=N*, *access_vlan=N*, *trunk_vlans=*

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*interface_mode*, *native_vlan*, *access_vlan*, *trunk_vlans*)

property xmlns

3.2.2 OSPFv2

ospfv2

class pytanga.components.OpenConfig.routing.ospfv2.ospfv2.**ospfv2Component**
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

property xmlns

ospfv2Area

class pytanga.components.OpenConfig.routing.ospfv2.ospfv2Area.**ospfv2AreaComponent**(*identifier*)
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*identifier*)

property xmlns

ospfv2Areas

class pytanga.components.OpenConfig.routing.ospfv2.ospfv2Areas.**ospfv2AreasComponent**
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

property xmlns

ospfv2Global

class pytanga.components.OpenConfig.routing.ospfv2.ospfv2Global.**ospfv2GlobalComponent**(*router_*
sum-
*mary_*
igp_sh
log_ad
hide_t

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*router_id*, *summary_route_cost_mode*, *igp_shortcuts*, *log_adjacency_changes*, *hide_transit_only_networks*)

property xmlns

ospfv2Interface

class pytanga.components.OpenConfig.routing.ospfv2.ospfv2Interface.**ospfv2InterfaceComponent**

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*if_id*, *network_type*, *priority*, *multi_area_adjacency_primary*, *authentication_type*, *metric*, *passive*, *hide_network*)

property xmlns

ospfv2Interfaces

class pytanga.components.OpenConfig.routing.ospfv2.ospfv2Interfaces.**ospfv2InterfacesCompone**

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

property xmlns

3.2.3 Static

interfaceref

class pytanga.components.OpenConfig.routing.static.interfaceref.**interfacerefComponent**(*interfa*
subin-
ter-
face=N

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*interface*, *subinterface*)

property xmlns

nexthop

class pytanga.components.OpenConfig.routing.static.nexthop.**nexthopComponent**(*index*,
next_hop=None,
met-
ric=None,
re-
curse=None)

Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*index*, *next_hop*, *metric*, *recurse*)

property xmlns

nexthops

class pytanga.components.OpenConfig.routing.static.nexthops.**nexthopsComponent**
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

property xmlns

static

class pytanga.components.OpenConfig.routing.static.static.**staticComponent**(*prefix*, *set_tag=None*, *description=None*, *operation=None*)
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

setAttributes(*prefix*, *set_tag*, *description*)

property xmlns

staticroutes

class pytanga.components.OpenConfig.routing.static.staticroutes.**staticroutesComponent**
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

property xmlns

3.3 AbstractComponent module

This modules define the Abstract Component Class

class pytanga.components.AbstractComponent.**AbstractComponent**
Bases: abc.ABC

abstract add(*component*) → None
This should method add a subComponent.

abstract getXMLNS()

abstract parse(*serializer:* pytanga.visitors.AbstractVisitor.AbstractVisitor)
This method should call the parse method for all childrens passing the serializer.

abstract remove(*component*) → None
This should method remove a subComponent.

3.4 Config module

Config component.

This module defines the config Component.

<config> </config>

class pytanga.components.config.**configComponent**
Bases: *pytanga.components.AbstractComponent.AbstractComponent*

add(*component*) → None
This should method add a subComponent.

getXMLNS()

is_composite() → bool

parse(*serializer*)
This method should call the parse method for all childrens passing the serializer.

remove(*component*) → None
This should method remove a subComponent.

property xmlns

CHAPTER

FOUR

HELPERS

4.1 Cisco IOS-XE

4.1.1 ConfigureBGP

class pytanga.helpers.Cisco.xe.bgp.**ConfigureBGP**(*asn*, *aigp_rib_metric=None*, *always_compare_med=None*, *cluster_id=None*, *deterministic_med=None*, *enforce_first_as=None*, *enhanced_error=None*, *fast_external_fallover=None*, *log_neighbor_changes=None*, *maxas_limit=None*, *maxcommunity_limit=None*, *route_map_cache=None*, *update_delay=None*, *router_id=None*)

Bases: object

addAfiSafiNeighbor(*afi_safi*, *address*, *activate=None*, *advertisement_interval=None*, *allow_policy=None*, *allowas_in=None*, *default_originate=None*, *default_originate_route_map=None*, *dmzlink_bw=None*, *maximum_prefix_n=None*, *maximum_prefix_threshold=None*, *maximum_prefix_restart=None*, *maximum_prefix_warning=None*, *next_hop_self=None*, *next_hop_self_all=None*, *next_hop_unchanged=None*, *route_reflector_client=None*, *send_community=None*, *send_label=None*, *soft_reconfiguration=None*, *weight=None*)

addAfiSafiNeighborRouteMap(*afi_safi*, *nei_address*, *inout*, *name*)

addAfiSafiNetwork(*afi_safi*, *network*, *mask=None*, *route_map=None*, *backdoor=None*)

addAfi_Safi(*afi_name*, *safi_name*)

addNeighbor(*address*, *remote_as=None*, *cluster_id=None*, *description=None*, *disable_connected_check=None*, *ebgp_multihop=None*, *password=None*, *peer_group=None*, *shutdown=None*, *keepalive_interval=None*, *holdtime=None*, *minimum_neighbor_hold=None*, *ttl_security=None*, *update_source=None*, *version=None*)

configureAfiSafiBGP(*afi_safi*, *advertise_best_external=None*, *dmzlink_bw=None*, *suppress_inactive=None*, *soft_reconfig_backup=None*, *scan_time=None*)

getBGP()

exception pytanga.helpers.Cisco.xe.bgp.**ConfigureBGPError**
Bases: Exception

4.1.2 ConfigurePrefixList

class pytanga.helpers.Cisco.xe.prefix.**ConfigurePrefixList**(*name*, *replace=False*, *step=5*)
Bases: object

Prefix List configuration helper Class

Parameters

- **name** (*string*) – the prefix list name
- **replace** (*string*) – set to config replace
- **step** (*integer*) – The sequence step for prefix list creation defaults 5

addPrefix(*action*, *network*, *seq=None*, *le=None*, *ge=None*)
Add a prefix to the prefix List

Parameters

- **action** (*string*) – The prefix action should be in [“permit” , “deny”]
- **network** (*string*) – The network
- **seq** (*integer, optional*) – The prefix sequence

getPrefixList()

Returns The prefixList Component

Return type *prefixeslistsComponent*

exception pytanga.helpers.Cisco.xe.prefix.**ConfigurePrefixListError**
Bases: Exception

4.2 OpenConfig

4.2.1 CreateIPInterface

pytanga.helpers.OpenConfig.interfaces.**createIPInterface**(*name*, *if_type*, *ip_version*, *address*, *prefix_length*, *if_description=None*, *if_mtu=None*, *enabled=None*, *sub_index=0*, *sub_desc=None*)

CHAPTER

FIVE

VISITORS

5.1 AbstractVisitor module

class pytanga.visitors.AbstractVisitor.**AbstractVisitor**
Bases: abc.ABC

abstract parse(*leaf*)

5.2 NETCONFvisitor module

class pytanga.visitors.netconfvisitor.**NETCONFVisitor**
Bases: *pytanga.visitors.AbstractVisitor.AbstractVisitor*

parse(*leaf*)

parseComponentData(*tag*, *data*, *xmlns*)

print(*output*)

exception pytanga.visitors.netconfvisitor.**VisitorError**
Bases: Exception

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

A

B

C

E

G

M

N

O

P

R

S

V

X