ntc-docs Documentation

Release 1.0

ntc

Oct 25, 2017

Contents

1 Introduction 3

2 Module Index 5

3 About Page 51

4 Frequently Asked Questions 53

Contents:

Hello and welcome to the docs!!!!!

CHAPTER 1

Introduction

Intro to NTC Documentation.

CHAPTER 2

Module Index

Arista Modules

These docs were dynamically created from the modules that can be found here.

eos_acl_entry

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.1.0

This module will manage standard ACL entries on EOS nodes

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.2 or later

Examples

```
- eos_acl_entry: seqno=10 name=foo action=permit srcaddr=0.0.0.0
  srcprefixlen=32

- eos_acl_entry: seqno=20 name=foo action=deny srcaddr=172.16.10.0
  srcprefixlen=16
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports stateful resource configuration.

eos_bgp_config

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.1.0

The eos_bgp_config module provides resource management of the global BGP routing process for Arista EOS nodes

Options

Important: Requires Arista EOS 4.13.7M or later with command API enable

Important: Requires Python Client for eAPI 0.4.0 or later

Examples

```
- name: enable BGP routing with AS 65535
  eos_bgp_config: bgp_as=65535 state=present enable=yes

- name: disable the BGP routing process
  eos_bgp_config: bgp_as=65535 enable=no
```

```
- name: configure the BGP router-id
  eos_bgp_config: bgp_as=65535 router_id=1.1.1.1

- name: configure the BGP with just max paths
  eos_bgp_config: bgp_as=65535 router_id=1.1.1.1 maximum_paths=20

- name: configure the BGP with maximum_paths and maximum_ecmp_paths
  eos_bgp_config: bgp_as=65535 router_id=1.1.1.1 maximum_paths=20
  maximum_ecmp_paths=20
```

Note: All configuraiton is idempontent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports tateful resource configuration

eos_bgp_neighbor

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.1.0

This eos_bgp_neighbor module provides stateful management of the neighbor statements for the BGP routing process for Arista EOS nodes

Options

Important: Requires Arista EOS 4.13.7M or later with command API enable

Important: Requires Python Client for eAPI 0.3.1 or later

Examples

```
- name: add neighbor 172.16.10.1 to BGP
  eos_bgp_neighbor: name=172.16.10.1 enable=yes remote_as=65000

- name: remove neighbor 172.16.10.1 to BGP
  eos_bgp_neighbor name=172.16.10.1 enable=yes remote_as=65000 state=absent
```

Note: All configuraiton is idempontent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports tateful resource configuration

eos_bgp_network

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.1.0

This eos_bgp_network module provides stateful management of the network statements for the BGP routing process for Arista EOS nodes

Options

Important: Requires Arista EOS 4.13.7M or later with command API enable

Important: Requires Python Client for eAPI 0.3.1 or later

Examples

```
- name: add network 172.16.10.0/26 with route-map test
  eos_bgp_network: prefix=172.16.10.0 masklen=26 route_map=test

- name: remove network 172.16.0.0/8
  eos_bgp_network: prefix=172.16.0.0 masklen=8 state=absent
```

Note: All configuraiton is idempontent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports tateful resource configuration

eos_command

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

The eos_command module provides a module for sending arbitray commands to the EOS node and returns the ouput. Only priviledged mode (enable) commands can be sent.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: execute show version and show hostname
  eos_command: commands='show version, show hostname'
```

Note: This module does not support idempotent operations.

Note: Supports eos metaparameters for using the eAPI transport

Note: This module does not support stateful configuration

eos_config

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

The eos_config module evalues the current configuration for specific commands. If the commands are either present or absent (depending on the function argument, the eos_config module will configure the node using the command argument.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: idempotent operation for removing a SVI
  eos_config:
    command='no interface Vlan100'
    regexp='interface Vlan100'
    state=absent

- name: non-idempotent operation for removing a SVI
  eos_config:
    command='no interface Vlan100'

- name: ensure default route is present
  eos_config:
    command='ip route 0.0.0.0/0 192.168.1.254'

- name: configure interface range to be shutdown if it isn't already
  eos_config:
    command='shutdown'
    regexp='(?<=[^no ] )shutdown'
    section='interface {{ item }}'
  with_items:
    - Ethernet1
    - Ethernet2
    - Ethernet3
```

Note: This module does not support idempotent operations.

Note: Supports eos metaparameters for using the eAPI transport

Note: This module does not support stateful configuration

eos_ethernet

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

The eos_ethernet module manages the interface configuration for physical Ethernet interfaces on EOS nodes.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: Ensure that Ethernet1/1 is administratively enabled
  eos_ethernet: name=Ethernet1/1 enable=yes

- name: Enable flowcontrol send and receive on Ethernet10
  eos_ethernet: name=Ethernet10 flowcontrol_send=yes flowcontrol_receive=yes
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Does not support stateful resource configuration.

eos_facts

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

The eos_facts module collects facts from the EOS for use in Ansible playbooks. It can be used independently as well to discover what facts are availble from the node. This facts module does not cache any facts. If no configuration options are specified, then all facts are returned.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: collect all facts from node
  eos_facts:

- name: include only a filtered set of facts returned
  eos_facts: include=interfaces

- name: exclude a specific set of facts
  eos_facts: exclude=vlans
```

Note: Supports eos metaparameters for using the eAPI transport

Note: The include and exclude options are mutually exclusive

eos_interface

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

The eos_interface module manages the interface configuration for any valid interface on EOS nodes.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: ensures the interface is configured
  eos_interface: name=Loopback0 state=present enable=yes

- name: ensures the interface is not configured
  eos_interface: name=Loopback1 state=absent
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports stateful resource configuration. This method also supports the 'default' state. This will default the specified interface. Note however that the default state operation is NOT idempotent.

eos_ipinterface

- *Synopsis*
- *Options*

- *Examples*

Synopsis

Added in version 1.0.0

The eos_ipinterface module manages logical layer 3 interface configurations.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: Ensure a logical IP interface is configured on Vlan100
  eos_ipinterface: name=Vlan100 state=present address=172.16.10.1/24

- name: Ensure a logical IP interface is not configured on Ethernet1
  eos_ipinterface: name=Ethernet1 state=absent

- name: Configure the MTU value on Port-Channel10
  eos_ipinterface: name=Port-Channel10 mtu=9000
```

Note: Currently this module only supports IPv4

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports stateful resource configuration.

eos_mlag_config

- *Synopsis*
- *Options*

- *Examples*

Synopsis

Added in version 1.0.0

The eos_mlag_interface module manages the MLAG interfaces on Arista EOS nodes. This module is fully stateful and all configuration of resources is idempotent unless otherwise specified.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: Ensure the MLAG domain-id is mlagPeer
  eos_mlag_config: domain_id=mlagPeer

- name: Configure the peer address and local interface
  eos_mlag_config: peer_address=2.2.2.2 local_interface=Vlan4094
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Does not support stateful resource configuration.

eos_mlag_interface

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

The eos_mlag_interface module manages the MLAG interfaces on Arista EOS nodes. This module is fully stateful and all configuration of resources is idempotent unless otherwise specified.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: Ensure Ethernet1 is configured with mlag id 10
  eos_mlag_interface: name=Ethernet1 state=present mlag_id=10

- name: Ensure Ethernet10 is not configured as mlag
  eos_mlag_interface: name=Ethernet10 state=absent
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports stateful resource configuration.

eos_ping

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

The eos_ping module will execute a network ping from the node and return the results. If the destination can be successfully pinged, then the module returns successfully. If any of the sent pings are not returned the module fails. By default, the error threshold is set to the same value as the number of pings sent

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.4.0 or later

Examples

```
- eos_ping: dst=192.168.1.254 count=10

# Set the error_threshold to 50% packet loss
- eos_ping: dst=192.168.1.254 count=10 error_threshold=50
```

Note: Important fixes to this module were made in pyeapi 0.4.0. Be sure to update to at least that version.

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Does not support stateful resource configuration.

eos_portchannel

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

The eos_portchannel module manages the interface configuration for logical Port-Channel interfaces on EOS nodes.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: Ensure Port-Channel1 has members Ethernet1 and 2
  eos_portchannel: name=Port-Channel1 members=Ethernet1,Ethernet2

- name: Ensure Port-Channel10 uses lacp mode active
  eos_portchannel: name=Port-Channel10 members=Ethernet1,Ethernet3
                   lacp_mode=active
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports stateful resource configuration.

eos_purge

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

The eos_purge module will scan the current nodes running-configuration and purge resources of a specified type if the resource is not explicitly configured in the playbook. This module will allow a playbook task to dynamically determine which resources should be removed from the nodes running-configuration based on the playbook. Note Purge is not supported for all EOS modules

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
# configure the set of vlans for the node

- name: configure vlans
  eos_vlan: vlanid={{ item }}
  with_items: ['1', '10', '11', '12', '13', '14', '15']
  register: required_vlans


# note the value for results is the registered vlan variable.  Also of
# importance is the to_nice_json filter which is required

- name: purge vlans not on the list
  eos_purge: resource=eos_vlan results='{{ required_vlans|to_nice_json }}'
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Does not support stateful resource configuration.

eos_routemap

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.2.0

This module will manage routemap entries on EOS nodes

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.4.0 or later

Examples

```
- eos_routemap: name=rm1 action=permit seqno=10
                description='this is a great routemap'
                match='as 50,interface Ethernet2'
                set='tag 100,weight 1000'
                continue=20
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports stateful resource configuration.

eos_staticroute

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.2.0

The eos_staticroute module manages static route configuration options on Arista EOS nodes.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.4.0 or later

Examples

```
- eos_staticroute: ip_dest=1.1.1.0/24 next_hop=Ethernet1
                   next_hop_ip=1.1.1.1 distance=1
                   tag=15 name=route1
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports stateful resource configuration.

eos_stp_interface

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

Provides active state management of STP interface configuration on Arista EOS nodes.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: Ensure portfast is enabled on Ethernet3
  eos_stp_interface: name=Ethernet3 portfast=yes

- name: Ensure bpduguard is enabled on Ethernet49
  eos_stp_interface: name=Ethernet49 bpduguard=yes
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Does not support stateful resource configuration.

eos_switchport

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

Provides active state management of switchport (layer 2) interface configuration in Arista EOS. Logical switchports are mutually exclusive with `eos_ipinterface`.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: Ensure Ethernet1 is an access port
  eos_switchport: name=Ethernet1 mode=access access_vlan=10

- name: Ensure Ethernet12 is a trunk port
  eos_switchport: name=Ethernet12 mode=trunk trunk_native_vlan=100

- name: Add the set of allowed vlans to Ethernet2/1
  eos_switchport: name=Ethernet2/1 mode=trunk trunk_allowed_vlans=1,10,100

- name: Add trunk group values to an interface
  eos_switchport: name=Ethernet5 trunk_groups=foo,bar,baz
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports stateful resource configuration.

eos_system

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

The eos_system module manages global system configuration options on Arista EOS nodes.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: configures the hostname to spine01
  eos_system: hostname=spine01
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports stateful resource configuration.

eos_user

- *Synopsis*

- *Options*
- *Examples*

Synopsis

Added in version 1.2.0

The eos_user module helps manage CLI users on your Arista nodes. You can create, delete and modify users along with their passwords.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.4.0 or later

Important: Requires Cleartext passwords are not accepted in playbooks

Examples

```
- name: Create simple user with no assigned password
  eos_user: name=simpletom nopassword=true

- name: Create user with MD5 password
  eos_user: name=securetom encryption=md5
            secret=$1$J0auuPhz$Pkr5NnHssW.Jqlk17Ylpk0

- name: Create user with SHA512 password (passwd truncated in eg)
  eos_user: name=securetom encryption=sha512
            secret=$6$somesalt$rkDq7Az4Efjo

- name: Remove user
  eos_user: name=securetom state=absent

- name: Create user with privilege level 10
  eos_user: name=securetom encryption=sha512
            secret=$6$somesalt$rkDq7Az4Efjo
            privilege=10

- name: Create user with role network-admin
  eos_user: name=securetom encryption=sha512
            secret=$6$somesalt$rkDq7Az4Efjo
            privilege=10 role=network-admin

- name: Add an SSH key with a user no password
  eos_user: name=sshkeytom nopassword=true
            sshkey='ssh-rsa somesshkey'
```

```
- name: Remove SSH key with a user no password
  eos_user: name=sshkeytom nopassword=true
            sshkey=''
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports stateful resource configuration.

eos_varp

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.2.0

This module will manage global Varp configuration on EOS nodes

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.4.0 or later

Examples

```
- eos_varp: mac_address='00:11:22:33:44:55'
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Does not support stateful resource configuration.

eos_varp_interface

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.2.0

This module will manage interface Varp configuration on EOS nodes. Typically this includes Vlan interfaces only by using the ip virtual-router address command.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.4.0 or later

Examples

```
- eos_varp_interface: name=Vlan1000 shared_ip='1.1.1.2,1.1.1.3,1.1.1.4'
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Does not support stateful resource configuration.

eos_vlan

- *Synopsis*

- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

The eos_vlan module manages VLAN configurations on Arista EOS nodes.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: ensures vlan 100 is configured
  eos_vlan: vlanid=100 state=present

- name: ensures vlan 200 is not configured
  eos_vlan: vlanid=200 state=absent

- name: configures the vlan name
  eos_vlan: vlanid=1 name=TEST_VLAN_1

- name: configure trunk groups for vlan 10
  eos_vlan: vlanid=10 trunk_groups=tg1,tg2,tg3
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports stateful resource configuration.

eos_vrrp

- *Synopsis*
- *Options*

- *Examples*

Synopsis

Added in version 1.2.0

This module will manage VRRP configurations on EOS nodes

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.4.0 or later

Examples

```
# Configure the set of tracked objects for the VRRP
# Create a list of dictionaries, where name is the object to be
# tracked, action is shutdown or decrement, and amount is the
# decrement amount. Amount is not specified when action is shutdown.

vars:
  tracks:
      - name: Ethernet1
        action: shutdown
      - name: Ethernet2
        action: decrement
        amount: 5

# Setup the VRRP

  - eos_vrrp:
      interface=Vlan70
      vrid=10
      enable=True
      primary_ip=10.10.10.1
      priority=50
      description='vrrp 10 on Vlan70'
      ip_version=2
      secondary_ip=['10.10.10.70','10.10.10.80']
      timers_advertise=15
      preempt=True
      preempt_delay_min=30
      preempt_delay_reload=30
      delay_reload=30
      track="{{ tracks }}"
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports stateful resource configuration.

eos_vxlan

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

The eos_vxlan module manages the logical VxLAN interface configuration on Arista EOS nodes.

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: ensures the vxlan interface is configured
  eos_vxlan: name=Vxlan1 state=present enable=yes

- name: ensures the vxlan interface is not configured
  eos_vxlan: name=Vxlan1 state=absent

- name: configures the vxlan source interface
  eos_vxlan: name=Vxlan1 source_interface=Loopback0
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports stateful resource configuration.

eos_vxlan_vlan

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

The eos_vxlan_vlan module manages the Vxlan VLAN to VNI mappings for an Arista EOS node that is operating as a VTEP

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: create a new vlan to vni mapping
  eos_vxlan_vlan: name=Vxlan1 state=present vlan=100 vni=1000

- name: remove an existing mapping if present in the config
  eos_vxlan_vlan: name=Vxlan1 state=absent vlan=200
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports stateful resource configuration.

eos_vxlan_vtep

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

The eos_vxlan_vtep module manages the Vxlan global VTEP flood list configure on Arista EOS nodes that are operating as VTEPs

Options

Important: Requires Arista EOS 4.13.7M or later with command API enabled

Important: Requires Python Client for eAPI 0.3.0 or later

Examples

```
- name: Ensures that 1.1.1.1 is in the global flood list
  eos_vxlan_vtep: name=Vxlan1 state=present vtep=1.1.1.1

- name: Ensures that 2.2.2.2 is not in the global flood list
  eos_vxlan_vtep: name=Vxlan1 state=absent vtep=2.2.2.2
```

Note: All configuration is idempotent unless otherwise specified

Note: Supports eos metaparameters for using the eAPI transport

Note: Supports stateful resource configuration.

Juniper Modules

These docs were dynamically created from the modules that can be found here.

junos_commit

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.2.0

Execute a Commit on a device running Junos independently of loading a configuration

Options

Important: Requires junos-eznc >= 1.2.2

Examples

```
- junos_commit:
   host: "{{ inventory_hostname }}"
   logfile=changes.log
   comment="Non load commit"
```

junos_get_config

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.2.0

Retrieve the configuration of a device running Junos and save it to a file. **Note** unicode chars will be converted to '??' as also done in PyEZ

Options

Important: Requires junos-eznc >= 1.2.2

Examples

```
- junos_get_config:
   host: "{{ inventory_hostname }}"
   logfile: get_config.log
   dest: "{{ inventory_hostname }}.xml"
   format: xml
   filter: "interfaces"
   options: {inherit: inherit, groups: groups}
```

junos_get_facts

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

Retrieve facts for a device running Junos OS, which includes information such as the serial number, product model, and Junos OS version. The module supports using both NETCONF and CONSOLE-based retrieval and returns the information as a JSON dictionary. The information is similar to facts gathered by other IT frameworks.

Options

Important: Requires junos-eznc >= 1.2.2

Important: Requires junos-netconify >= 1.0.1, when using the *console* option

Examples

```
# retrieve facts using NETCONF, assumes ssh-keys

- junos_get_facts: host={{ inventory_hostname }}
  register: junos

# retrieve facts using CONSOLE, assumes Amnesiac system
# root login, no password

- junos_get_facts:
    host={{ inventory_hostname }}
    user=root
    console="--telnet={{TERMSERV}},{{TERMSERVPORT}}"
    savedir=/usr/local/junos/inventory
```

```
  register: junos

# access the facts

- name: version
  debug: msg="{{ junos.facts.version }}"
```

junos_install_config

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

Load a complete Junos OS configuration (overwrite) or merge a configuration snippet onto a device running Junos OS and commit it. The default behavior is to perform a **load merge** operation (overwrite='no'). This module performs an atomic lock/edit/unlock. If the process fails at any step, then all configuration changes are discarded. You can load the configuration using either NETCONF or the CONSOLE port. Specify the *console* option to use the CONSOLE port. You provide the configuration data in a file. Supported formats when using NETCONF include ASCII text, Junos XML elements, and Junos OS **set** commands. Configurations performed through the console must only use ASCII text formatting.

Options

Important: Requires junos-eznc >= 1.2.2

Important: Requires junos-netconify >= 1.0.1, when using the *console* option

Examples

```
# load merge a change to the Junos OS configuration using NETCONF

- junos_install_config:
    host={{ inventory_hostname }}
    file=banner.conf

# load overwrite a new Junos OS configuration using the CONSOLE port

- junos_install_config:
    host={{ inventory_hostname }}
    console="--telnet={{TERMSERV}},{{TERMSERV_PORT}}"
```

```
    file=default_new_switch.conf
    overwrite=yes

# load merge a change to the Junos OS configuration using NETCONF and supplying a␣
↪commit log message
- junos_install_config:
    host={{ inventory_hostname }}
    file=banner.conf
    comment="configured by ansible"

# load replace a change to the Junos OS configuration using NETCONF
- junos_install_config:
    host={{ inventory_hostname }}
    file=snmp.conf
    replace=yes
```

junos_install_os

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

Install a Junos OS image on one or more Routing Engines. This module supports installations on single Routing Engine devices, MX Series routers with dual Routing Engines, and EX Series switches in a non-mixed Virtual Chassis. This action is equivalent to performing the Junos OS **request system software add** operational command. If the existing Junos OS version matches the desired version, no action is performed, and the "changed" attribute reports False. If the existing version does not match, then the module performs the following actions (1) Computes the MD5 checksum of the package located on the server. (2) Copies the Junos OS software package to the device running Junos OS. (3) Computes the MD5 checksum on the device running Junos OS and compares the two. (4) Installs the Junos OS software package. (5) Reboots the device (default). Running the module in check mode reports whether the current Junos OS version matches the desired version.

Options

Important: Requires py-junos-eznc >= 1.2.2

Examples

```
- junos_install_os:
    host={{ inventory_hostname }}
    version=12.1X46-D10.2
```

```
    package=/usr/local/junos/images/junos-vsrx-12.1X46-D10.2-domestic.tgz
    logfile=/usr/local/junos/log/software.log
```

junos_rollback

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.2.0

Rollback the configuration of a device running Junos

Options

Important: Requires junos-eznc >= 1.2.2

Examples

```
- junos_rollback:
   host: "{{ inventory_hostname }}"
   logfile=rollback.log
   diffs_file=rollback.diff
   rollback=1
   comment="Rolled back by Ansible"
   confirm=5
```

junos_shutdown

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

Shut down (power off) or reboot a device running Junos OS. This includes all Routing Engines in a Virtual Chassis or a dual Routing Engine system. This is equivalent to executing either the Junos OS **request system power-off** or **request system reboot** operational command.

Options

Important: Requires junos-eznc >= 1.2.2

Examples

```
- junos_shutdown:
    host={{ inventory_hostname }}
    shutdown="shutdown"
    reboot=yes
```

junos_srx_cluster

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.2.0

Create an srx chassis cluster and reboot the device. The device must be capable of forming an srx cluster and have the correct cables installed.

Options

Important: Requires junos-eznc >= 1.2.2

Examples

```
-junos_srx_cluster:
  host={{ inventory_hostname }}
  console="--port={{ serial }}"
  user=rick
  passwd=password123
  cluster_enable=true
  logfile=cluster.log
  cluster_id={{ cluster_id }}
```

```
  node={{ node_id }}

-junos_srx_cluster:
  host={{ inventory_hostname }}
  user=rick
  passwd=password123
  cluster_enable=false
  logfile=cluster.log
```

junos_zeroize

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

Execute the Junos OS **request system zeroize** command to remove all configuration information on the Routing Engines and reset all key values on a device running Junos OS. The command removes all data files, including customized configuration and log files, by unlinking the files from their directories. The command also removes all user-created files from the system including all plain-text passwords, secrets, and private keys for SSH, local encryption, local authentication, IPsec, RADIUS, TACACS+, and SNMP. This command reboots the device and sets it to the factory default configuration. After the reboot, you must log in through the console as root in order to access the device.

Options

Important: Requires junos-eznc >= 1.2.2

Important: Requires junos-netconify >= 1.0.1, when using the *console* option

Examples

```
- junos_zeroize:
    host={{ inventory_hostname }}
    zeroize="zeroize"
```

Note: You **MUST** either use the *host* option or the *console* option to designate how the device is accessed.

ntc-ansible Modules (multi-vendor)

These docs were dynamically created from the modules that can be found here.

compare_dict

- *Synopsis*
- *Options*
- *Examples*

Synopsis

This module verifies that the result received from TextFSM for a particular template matches the expected output from a test scenario. It does so by comparing two lists of dictionaries going through elements of one and checking if the element 'is in' the second list.

Options

Important: Requires none

Examples

```
# verify that parsed result is the same as expected
- compare_dict:
    result: "{{ item.item.response }}"
    sample: "{{ item.ansible_facts.parsed_sample }}"
```

get_test_info

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Offers ability to dynamically create a list of dictionaries with info required to test all templates. This will loop through the tests dir and build each dictionary to have command, platform, rawfile, parsedfile, and path for each.

Options

Examples

```
- get_test_info:
```

ntc_config_command

- *Synopsis*
- *Options*
- *Examples*

Synopsis

This module write config data to devices that don't have an API. The use case would be writing configuration based on output gleaned from ntc_show_command output.

Options

Important: Requires netmiko

Examples

```
# write vlan data
- ntc_config_command:
    connection: ssh
    platform: cisco_nxos
    commands:
      - vlan 10
      - name vlan_10
      - end
    host: "{{ inventory_hostname }}"
    username: "{{ username }}"
    password: "{{ password }}"
    secret: "{{ secret }}"

# write config from file
- ntc_config_command:
    connection: ssh
    platform: cisco_nxos
    commands_file: "dynamically_created_config.txt"
    host: "{{ inventory_hostname }}"
    username: "{{ username }}"
    password: "{{ password }}"
    secret: "{{ secret }}"
```

ntc_file_copy

- *Synopsis*
- *Options*
- *Examples*
- *Return Values*

Synopsis

Added in version 1.9.2

Copy a file to the flash (or bootflash) remote network device on supported platforms over SCP. Supported platforms include Cisco Nexus switches with NX-API, Cisco IOS switches or routers, Arista switches with eAPI.

Options

Important: Requires pyntc

Examples

```
- ntc_file_copy:
    platform: cisco_nxos_nxapi
    local_file: /path/to/file
    host: "{{ inventory_hostname }}"
    username: "{{ username }}"
    password: "{{ password }}"
    transport: http
- ntc_file_copy:
    ntc_host: n9k1
    ntc_conf_file: .ntc.conf
    local_file: /path/to/file
- ntc_file_copy:
    ntc_host: eos_leaf
    local_file: /path/to/file
- ntc_file_copy:
    platform: arista_eos_eapi
    local_file: /path/to/file
    remote_file: /path/to/remote_file
    host: "{{ inventory_hostname }}"
    username: "{{ username }}"
    password: "{{ password }}"
- ntc_file_copy:
    platform: cisco_ios
    local_file: "{{ local_file_1 }}"
    host: "{{ inventory_hostname }}"
    username: "{{ username }}"
    password: "{{ password }}"
    secret: "{{ secret }}"
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Note: On NXOS, the feature must be enabled with feature scp-server.

Note: On IOS and Arista EOS, the user must be at privelege 15.

Note: If the file is already present (md5 sums match), no transfer will take place.

Note: Check mode will tell you if the file would be copied.

Note: The same user credentials are used on the API/SSH channel and the SCP file transfer channel.

ntc_get_facts

- *Synopsis*
- *Options*
- *Examples*
- *Return Values*

Synopsis

Added in version 1.9.2

Reboot a network device, optionally on a timer. Supported platforms include Cisco Nexus switches with NX-API, Cisco IOS switches or routers, Arista switches with eAPI.

Options

Important: Requires pyntc

Examples

```
- ntc_get_facts:
    platform: cisco_nxos_nxapi
    host: "{{ inventory_hostname }}"
    username: "{{ username }}"
```

```
    password: "{{ password }}"
    transport: http

- ntc_get_facts:
    ntc_host: n9k1
    ntc_conf_file: .ntc.conf

- ntc_get_facts:
    ntc_host: eos_leaf

- ntc_get_facts:
    platform: arista_eos_eapi
    host: "{{ inventory_hostname }}"
    username: "{{ username }}"
    password: "{{ password }}"

- ntc_get_facts:
    platform: cisco_ios
    host: "{{ inventory_hostname }}"
    username: "{{ username }}"
    password: "{{ password }}"
    secret: "{{ secret }}"
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Note: Facts to be returned include - uptime (string), uptime (seconds), model, vendor, os_version, serial_number, hostname, fqdn, vlans, interfaces.

Note: Facts are automatically added to Ansible facts environment. No need to register them.

ntc_install_os

- *Synopsis*
- *Options*
- *Examples*
- *Return Values*

Synopsis

Added in version 1.9.2

Set boot options like boot image and kickstart image. Supported platforms include Cisco Nexus switches with NX-API, Cisco IOS switches or routers, Arista switches with eAPI.

Options

Important: Requires pyntc

Examples

```
- ntc_install_os:
    ntc_host: n9k1
    system_image_file: n9000-dk9.6.1.2.I3.1.bin

- ntc_install_os:
    ntc_host: n3k1
    system_image_file: n3000-uk9.6.0.2.U6.5.bin
    kickstart_image_file: n3000-uk9-kickstart.6.0.2.U6.5.bin

- ntc_install_os:
    ntc_host: c2801
    system_image_file: c2800nm-adventerprisek9_ivs_li-mz.151-3.T4.bin
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Note: Do not include full file paths, just the name of the file(s) stored on the top level flash directory.

Note: You must know if your platform supports taking a kickstart image as a parameter. If supplied but not supported, errors may occur.

Note: It may be useful to use this module in conjuction with ntc_file_copy and ntc_reboot.

Note: With NXOS devices, this module attempts to install the software immediately, wich may trigger a reboot.

Note: With NXOS devices, install process may take up to 10 minutes, especially if the device reboots.

Note: Tested on Nexus 3000, 5000, 9000.

Note: In check mode, the module tells you if the current boot images are set to the desired images.

ntc_reboot

- *Synopsis*
- *Options*
- *Examples*
- *Return Values*

Synopsis

Added in version 1.9.2

Reboot a network device, optionally on a timer. Supported platforms include Cisco Nexus switches with NX-API, Cisco IOS switches or routers, Arista switches with eAPI.

Options

Important: Requires pyntc

Examples

```
- ntc_reboot:
    platform: cisco_nxos_nxapi
    confirm: true
    host: "{{ inventory_hostname }}"
    username: "{{ username }}"
    password: "{{ password }}"
    transport: http

- ntc_reboot:
    ntc_host: n9k1
    ntc_conf_file: .ntc.conf
    confirm: true

- ntc_file_copy:
    platform: arista_eos_eapi
    confirm: true
    host: "{{ inventory_hostname }}"
    username: "{{ username }}"
    password: "{{ password }}"

- ntc_file_copy:
    platform: cisco_ios
    confirm: true
    timer: 5
    host: "{{ inventory_hostname }}"
    username: "{{ username }}"
    password: "{{ password }}"
    secret: "{{ secret }}"
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

ntc_rollback

- *Synopsis*
- *Options*
- *Examples*
- *Return Values*

Synopsis

This module offers the ability to set a configuration checkpoint file or rollback to a configuration checkpoint file on supported Cisco or Arista switches. Supported platforms include Cisco Nexus switches with NX-API, Cisco IOS switches or routers, Arista switches with eAPI.

Options

Important: Requires pyntc

Examples

```
- ntc_rollback:
    ntc_host: eos1
    checkpoint_file: backup.cfg

- ntc_rollback:
    ntc_host: eos1
    rollback_to: backup.cfg
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Note: This module is not idempotent.

ntc_save_config

- *Synopsis*
- *Options*
- *Examples*
- *Return Values*

Synopsis

Added in version 1.9.2

Save the running configuration as the startup configuration or to a file on the network device. Optionally, save the running configuration to this computer. Supported platforms include Cisco Nexus switches with NX-API, Cisco IOS switches or routers, Arista switches with eAPI.

Options

Important: Requires pyntc

Examples

```
- ntc_save_config:
    platform: cisco_nxos_nxapi
    host: "{{ inventory_hostname }}"
    username: "{{ username }}"
    password: "{{ password }}"

- ntc_save_config:
    ntc_host: n9k1

- ntc_save_config:
    platform: arista_eos_eapi
    host: "{{ inventory_hostname }}"
    username: "{{ username }}"
    password: "{{ password }}"
    remote_file: running_config_copy.cfg
    transport: https

# You can get the timestamp by setting get_facts to True, then you can append it to␣
↪your filename.

- ntc_save_config:
    platform: cisco_ios
    host: "{{ inventory_hostname }}"
    username: "{{ username }}"
    password: "{{ password }}"
    local_file: config_{{ inventory_hostname }}_{{ ansible_date_time.date | replace('-
↪','_') }}.cfg
```

Return Values

Common return values are documented here common_return_values, the following are the fields unique to this module:

Note: This module is not idempotent.

ntc_show_command

- *Synopsis*
- *Options*
- *Examples*

Synopsis

This module offers structured data for CLI enabled devices by using the TextFSM library for templating and netmiko for SSH connectivity

Options

Important: Requires netmiko

Important: Requires textfsm

Important: Requires terminal

Examples

```
# get vlan data
- ntc_show_command:
    connection=ssh
    platform=cisco_nxos
    command='show vlan'
    host={{ inventory_hostname }}
    username={{ username }}
    password={{ password }}

# get snmp community
- ntc_show_command:
    connection=ssh
    platform=cisco_nxos
    command='show snmp community'
    host={{ inventory_hostname }}
```

```
    username={{ username }}
    password={{ password }}
    secret:{{ secret }}
```

ansible-snmp Modules (mult-vendor)

These docs were dynamically created from the modules that can be found here.

snmp_device_version

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Returns vendor, os and version of an SNMP device.

Options

Important: Requires nelsnmp

Examples

```
# Get device info with SNMPv2
- snmp_device_version: host={{ inventory_hostname }} version=2c community=public

# Get device info with SNMPv3
- cisco_snmp_save_config:
    host={{ inventory_hostname }}
    version=3
    level=authPriv
    integrity=sha
    privacy=aes
    username=snmp-user
    authkey=abc12345
    privkey=def6789
```

napalm Modules (mult-vendor)

These docs were dynamically created from the modules that can be found here.

These docs were dynamically created from the modules that can be found here.

napalm_install_config

- *Synopsis*
- *Options*
- *Examples*

Synopsis

Added in version 1.0.0

Gathers facts from a network device

Options

Important: Requires napalm

Examples

```
- napalm_install_config:
    hostname={{ inventory_hostname }}
    username=dbarroso
    dev_os={{os}}
    password=p4ssw0rd
```

CHAPTER 3

About Page

About these modules

CHAPTER 4

Frequently Asked Questions

Here are some commonly-asked questions and their answers.

This is question number 1?

This is the answer to question 1, do code blocks like this:

```
ntc-ansible:
  command: fake command
```

This is question number 2?

The answer to this question doesn't use a code block.