Junos Ansible Modules Documentation

Release 2.4.0

Juniper Networks, Inc.

Jul 23, 2020

Contents

Contents:

CHAPTER 1

juniper_junos_facts

Retrieve facts from a Junos device

New in version 2.0.0.

1.1 Synopsis

- Retrieve facts from a Junos device using the PyEZ fact gathering system.
- Also returns the committed configuration of the Junos device if the *config_format* option has a value other than `none`.

1.2 Requirements

The following software packages must be installed on hosts that execute this module:

- junos-eznc >= 2.2.0
- Python >= 2.7

1.3 Module-specific Options

The following options may be specified for this module:

1.4 Common Connection-related Options

In addition to the *Module-specific Options*, the following connection-related options are also supported by this module:

1.5 Common Logging-related Options

In addition to the *Module-specific Options*, the following logging-related options are also supported by this module:

1.6 Examples

```
---
- name: Gather facts from Junos devices
  hosts: junos-all
  connection: local
  gather_facts: no
  roles:
    - Juniper.junos
  tasks:
    - name: Gather Junos facts with no configuration
      juniper_junos_facts:

# Print a fact

# Using config_format option

# Print the config

# Using savedir option

# Print the saved JSON file
```

1.7 Return Values

1.8 Notes

Note:

- The NETCONF system service must be enabled on the target Junos device.

1.8.1 Author

- Juniper Networks - Stacy Smith (@stacywsmith)

1.8.2 Status

This module is flagged as **stableinterface** which means that the maintainers for this module guarantee that no backward incompatible interface changes will be made.

CHAPTER 2

juniper_junos_command

Execute one or more CLI commands on a Junos device

New in version 2.0.0.

2.1 Synopsis

- Execute one or more CLI commands on a Junos device.
- This module does NOT use the Junos CLI to execute the CLI command. Instead, it uses the `<command>` RPC over a NETCONF channel. The `<command>` RPC takes a CLI command as it's input and is very similar to executing the command on the CLI, but you can NOT include any pipe modifies (i.e. `| match`, `| count`, etc.) with the CLI commands executed by this module.

2.2 Requirements

The following software packages must be installed on hosts that execute this module:

- junos-eznc >= 2.2.0
- Python >= 2.7

2.3 Module-specific Options

The following options may be specified for this module:

2.4 Common Connection-related Options

In addition to the *Module-specific Options*, the following connection-related options are also supported by this module:

2.5 Common Logging-related Options

In addition to the *Module-specific Options*, the following logging-related options are also supported by this module:

2.6 Examples

```
---
- name: Examples of juniper_junos_command
  hosts: junos-all
  connection: local
  gather_facts: no
  roles:
    - Juniper.junos

  tasks:
    - name: Execute single "show version" command.
      juniper_junos_command:
        commands: "show version"
      register: response

    - name: Print the command output
      debug:
        var: response.stdout

    - name: Execute three commands.
      juniper_junos_command:
        commands:
          - "show version"
          - "show system uptime"
          - "show interface terse"
      register: response

    - name: Print the command output of each.
```

(continues on next page)

(continued from previous page)

```
    debug:
      var: item.stdout
    with_items: "{{ response.results }}"

  - name: Two commands with XML output.
    juniper_junos_command:
      commands:
        - "show route"
        - "show lldp neighbors"
      format: xml

  - name: show route with XML output - show version with JSON output
    juniper_junos_command:
      commands:
        - "show route"
        - "show version"
      formats:
        - "xml"
        - "json"

  - name: save outputs in dest_dir
    juniper_junos_command:
      commands:
        - "show route"
        - "show version"
      dest_dir: "./output"

  - name: save output to dest
    juniper_junos_command:
      command: "show system uptime"
      dest: "/tmp/{{ inventory_hostname }}.uptime.output"

  - name: save output to dest
    juniper_junos_command:
      command:
        - "show route"
        - "show lldp neighbors"
      dest: "/tmp/{{ inventory_hostname }}.commands.output"

  - name: Multiple commands, save outputs, but don't return them
    juniper_junos_command:
      commands:
        - "show route"
        - "show version"
      formats:
        - "xml"
        - "json"
      dest_dir: "/tmp/outputs/"
      return_output: false
```

2.7 Return Values

2.8 Notes

Note:

- The NETCONF system service must be enabled on the target Junos device.

2.8.1 Author

- Juniper Networks - Stacy Smith (@stacywsmith)

2.8.2 Status

This module is flagged as **stableinterface** which means that the maintainers for this module guarantee that no backward incompatible interface changes will be made.

CHAPTER 3

juniper_junos_software

Install software on a Junos device

3.1 Synopsis

- Install a Junos OS image, or other software package, on a Junos device. This action is generally equivalent to the `request system software add` operational-mode CLI command. It performs the following steps in order:

1. Compare the currently installed Junos version to the desired version specified by the *version* option.
   - If the current and desired versions are the same, stop and return *changed* with a value of `false`.
   - If running in check mode, and the current and desired versions differ, stop and return *changed* with a value of `true`.

- Otherwise, proceed.

2. If the *local_package* option is specified, compute the MD5 checksum of the *local_package* file on the local Ansible control machine.
3. Check if the file exists at the *remote_package* location on the target Junos device. If so, compute the MD5 checksum of the file on the target Junos device.
4. If the *cleanfs* option is `true`, the default, then perform the equivalent of the `request system storage cleanup` CLI command.
5. If the checksums computed in steps 2 and 3 differ, or if the *remote_package* file does not exist on the target Junos device, then copy the package from *local_package* on the local Ansible control machine to *remote_package* on the target Junos device.
6. Install the software pacakge from the *remote_package* location on the target Junos device using the options specified.
7. If the *reboot* option is `true`, the default, initiate a reboot of the target Junos device.

3.2 Requirements

The following software packages must be installed on hosts that execute this module:

- junos-eznc >= 2.2.0
- Python >= 2.7

3.3 Module-specific Options

The following options may be specified for this module:

3.4 Common Connection-related Options

In addition to the *Module-specific Options*, the following connection-related options are also supported by this module:

3.5 Common Logging-related Options

In addition to the *Module-specific Options*, the following logging-related options are also supported by this module:

3.6 Examples

```
---
- name: Examples of juniper_junos_software
  hosts: junos-all
  connection: local
  gather_facts: no
  roles:
    - Juniper.junos
```

(continues on next page)

(continued from previous page)

```
  tasks:
    - name: Execute a basic Junos software upgrade.
      juniper_junos_software:
        local_package: "./images/"
      register: response
    - name: Print the complete response.
      debug:
        var: response

###### OLD 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
###### OLD EXAMPLES ##########
```

3.7 Return Values

3.8 Notes

Note:

- This module does support connecting to the console of a Junos device, but does not support copying the software package from the local Ansible control machine to the target Junos device while connected via the console. In this situation, the *remote_package* option must be specified, and the specified software package must already exist on the target Junos device.
- This module returns after installing the software and, optionally, initiating a reboot of the target Junos device. It does not wait for the reboot to complete, and it does not verify that the desired version of software specified by the *version* option is actually activated on the target Junos device. It is the user's responsibility to confirm the software installation using additional follow on tasks in their playbook.
- The NETCONF system service must be enabled on the target Junos device.

3.8.1 Author

- Jeremy Schulman
- Juniper Networks - Stacy Smith (@stacywsmith)

3.8.2 Status

This module is flagged as **stableinterface** which means that the maintainers for this module guarantee that no backward incompatible interface changes will be made.

CHAPTER 4

juniper_junos_jsnapy

Execute JSNAPy tests on a Junos device

New in version 2.0.0.

4.1 Synopsis

- Execute Junos SNAPshot Adminsitrator (JSNAPy) tests against a Junos device. JSNAPy is documented on Github and this Day One Book
- This module only reports `failed` if the module encounters an error and fails to execute the JSNAPy tests. If does NOT report `failed` if one or more of the JSNAPy tests fail. To check the test results, register the module's response and use the assert module to verify the expected result in the response. (See *Examples*.)
- A callback plugin which formats and prints JSNAPy test results for human consumption is also available. This callback plugin is enabled by adding `callback_whitelist = jsnapy` to the Ansible configuration file.

4.2 Requirements

The following software packages must be installed on hosts that execute this module:

- junos-eznc >= 2.2.0
- Python >= 2.7

4.3 Module-specific Options

The following options may be specified for this module:

4.4 Common Connection-related Options

In addition to the *Module-specific Options*, the following connection-related options are also supported by this module:

4.5 Common Logging-related Options

In addition to the *Module-specific Options*, the following logging-related options are also supported by this module:

4.6 Examples

```
---
- name: Examples of juniper_junos_jsnapy
  hosts: junos-all
  connection: local
  gather_facts: no
  roles:
    - Juniper.junos

  tasks:
    - name: JUNOS Post Checklist
      juniper_junos_jsnapy:
        action: "snap_post"
        config_file: "first_test.yml"
        logfile: "migration_post.log"
      register: test1
    - name: Verify all JSNAPy tests passed
      assert:
        that:
          - "test1.passPercentage == 100"
    - name: Print the full test response
      debug:
        var: test1

    - name: Test based on a test_file directly
      juniper_junos_jsnapy:
       action: "snapcheck"
       test_files: "tests/test_junos_interface.yaml"
```

(continues on next page)

(continued from previous page)

```
    register: test2
  - name: Verify all JSNAPy tests passed
    assert:
      that:
        - "test2.passPercentage == 100"
  - name: Print the full test response
    debug:
      var: test2

  - name: "Collect Pre Snapshot"
    juniper_junos_jsnapy:
      action: "snap_pre"
      test_files: "tests/test_loopback.yml"

  - name: "Collect Post Snapshot"
    juniper_junos_jsnapy:
      action: "snap_post"
      test_files: "tests/test_loopback.yml"

  - name: "Check after Pre and Post Snapshots"
    juniper_junos_jsnapy:
      action: "check"
      test_files: "tests/test_loopback.yml"
    register: test3
  - name: Verify all JSNAPy tests passed
    assert:
      that:
        - "test3.|succeeded"
        - "test3.passPercentage == 100"
  - name: Print the full test response
    debug:
      var: test3
```

4.7 Return Values

4.8 Notes

Note:

- The NETCONF system service must be enabled on the target Junos device.

4.8.1 Author

- Juniper Networks
- Roslan Zaki
- Damien Garros
- Stacy Smith (@stacywsmith)”

4.8.2 Status

This module is flagged as **stableinterface** which means that the maintainers for this module guarantee that no backward incompatible interface changes will be made.

CHAPTER 5

juniper_junos_config

Manipulate the configuration of a Junos device

New in version 2.0.0.

5.1 Synopsis

- Manipulate the configuration of a Junos device. This module allows a combination of loading or rolling back, checking, diffing, retrieving, and committing the configuration of a Junos device. It performs the following steps in order:

1. Open a candidate configuration database.
   - If the *config_mode* option has a value of `exclusive`, the default, take a lock on the candidate configuration database. If the lock fails the module fails and reports an error.

- If the *config_mode* option has a value of `private`, open a private candidate configuration database. If opening the private configuration database fails the module fails and reports an error.

2. Load configuration data into the candidate configuration database.
   - Configuration data may be loaded using the *load* or *rollback* options. If either of these options are specified, new configuration data is loaded. If neither option is specified, this step is skipped.
   - If the *rollback* option is specified, replace the candidate configuration with the previous configuration specified by the value of the *rollback* option.
   - If the *load* option is specified, load new configuration data.
   - The value of the *load* option defines the type of load which is performed.
   - The source of the new configuration data is one of the following:
     - *src* - A file path on the local Ansible control machine.
     - *lines* - A list of strings containing the configuration data.
     - *template* - A file path to a Jinja2 template on the local Ansible control machine. This template is rendered with the variables specified by the *vars* option. If the *template* option is specified, the *vars* option must also be specified.
     - *url* - A URL reachable from the target Junos device.
   - If the *format* option is specified, the configuration file being loaded is in the specified format, rather than the format determined from the file name.
3. Check the validity of the candidate configuration database.
   - If the *check* option is `true`, the default, check the validity of the configuration by performing a "commit check" operation.
   - This option may be specified with *diff* `false` and *commit* `false` to confirm a previous "commit confirmed <min>" operation without actually performing an additional commit.
   - If the configuration check fails, further processing stops, the module fails, and an error is reported.
4. Determine differences between the candidate and committed configuration databases.
   - If step 2 was not skipped, and the *diff* option is `true`, the default, perform a diff between the candidate and committed configuration databases.
   - If the *diffs_file* or *dest_dir* option is specified, save the generated configuration differences.
   - If the *return_output* option is `true`, the default, include the generated configuration difference in the *diff* and *diff_lines* keys of the module's response.
5. Retrieve the configuration database from the Junos device.
   - If the *retrieve* option is specified, retrieve the configuration database specified by the *retrieve* value from the target Junos device to the local Ansible control machine.
   - The format in which the configuration is retrieved is specified by the value of the *format* option.
   - The optional *filter* controls which portions of the configuration are retrieved.
   - If *options* are specified, they control the content of the configuration retrieved.
   - If the *dest* or *dest_dir* option is specified, save the retrieved configuration to a file on the local Ansible control machine.
   - If the *return_output* option is `true`, the default, include the retrieved configuration in the *config*, *config_lines*, and *config_parsed* keys of the module's response.

6. Commit the configuration changes.
   - If the *commit* option is `true`, the default, commit the configuration changes.
   - This option may be specified with *diff* `false` and *check* `false` to confirm a previous "commit confirmed <min>" operation.
   - If the *comment* option is specified, add the comment to the commit.
   - If the *confirmed* option is specified, perform a `commit confirmed` *min* operation where *min* is the value of the *confirmed* option.
   - If the *check* option is `true` and the *check_commit_wait* option is specified, wait *check_commit_wait* seconds before performing the commit.
7. Close the candidate configuration database.
   - Close and discard the candidate configuration database.
   - If the *config_mode* option has a value of `exclusive`, the default, unlock the candidate configuration database.

5.2 Requirements

The following software packages must be installed on hosts that execute this module:

- junos-eznc >= 2.2.0
- Python >= 2.7

5.3 Module-specific Options

The following options may be specified for this module:

5.4 Common Connection-related Options

In addition to the *Module-specific Options*, the following connection-related options are also supported by this module:

5.5 Common Logging-related Options

In addition to the *Module-specific Options*, the following logging-related options are also supported by this module:

5.6 Examples

```
---
- name: Manipulate the configuration of Junos devices
  hosts: junos-all
  connection: local
  gather_facts: no
  roles:
```

(continues on next page)

(continued from previous page)

```
    - Juniper.junos
  tasks:
    - name: Retrieve the committed configuration
      juniper_junos_config:
        retrieve: 'committed'
        diff: false
        check: false
        commit: false
      register: response
    - name: Print the lines in the config.
      debug:
        var: response.config_lines

    - name: Append .foo to the hostname using private config mode.
      juniper_junos_config:
        config_mode: 'private'
        load: 'merge'
        lines:
          - "set system host-name {{ inventory_hostname }}.foo"
      register: response
    - name: Print the config changes.
      debug:
        var: response.diff_lines

    - name: Rollback to the previous config.
      juniper_junos_config:
        config_mode: 'private'
        rollback: 1
      register: response
    - name: Print the config changes.
      debug:
        var: response.diff_lines

    - name: Rollback to the rescue config.
      juniper_junos_config:
        rollback: 'rescue'
      register: response
    - name: Print the complete response.
      debug:
        var: response

    - name: Load override from a file.
      juniper_junos_config:
        load: 'override'
        src: "{{ inventory_hostname }}.conf"
      register: response
    - name: Print the complete response.
      debug:
        var: response

    - name: Load from a Jinja2 template.
      juniper_junos_config:
        load: 'merge'
        format: 'xml'
        template: "{{ inventory_hostname }}.j2"
        vars:
          host: "{{ inventory_hostname }}"
```

(continues on next page)

(continued from previous page)

```
    register: response
  - name: Print the complete response.
    debug:
      var: response

  - name: Load from a file on the Junos device.
    juniper_junos_config:
      load: 'merge'
      url: "{{ inventory_hostname }}.conf"
    register: response
  - name: Print the complete response.
    debug:
      var: response

  - name: Load from a file on the Junos device, skip the commit check
    juniper_junos_config:
      load: 'merge'
      url: "{{ inventory_hostname }}.conf"
      check: false
    register: response
  - name: Print the msg.
    debug:
      var: response.msg

  - name: Print diff between current and rollback 10. No check. No commit.
    juniper_junos_config:
      rollback: 11
      diff: true
      check: false
      commit: false
    register: response
  - name: Print the msg.
    debug:
      var: response

  - name: Retrieve [edit system services] of current committed config.
    juniper_junos_config:
      retrieve: 'committed'
      filter: 'system/services'
      diff: true
      check: false
      commit: false
    register: response
  - name: Print the resulting config lines.
    debug:
      var: response.config_lines

  - name: Enable NETCONF SSH and traceoptions, save config, and diffs.
    juniper_junos_config:
      load: 'merge'
      lines:
        - 'set system services netconf ssh'
        - 'set system services netconf traceoptions flag all'
        - 'set system services netconf traceoptions file netconf.log'
      format: 'set'
      retrieve: 'candidate'
      filter: 'system/services'
```

(continues on next page)

(continued from previous page)

```
      comment: 'Enable NETCONF with traceoptions'
      dest_dir: './output'
    register: response
  - name: Print the complete response
    debug:
      var: response

  - name: Load conf. Confirm within 5 min. Wait 3 secs between chk and commit
    juniper_junos_config:
      load: 'merge'
      url: "{{ inventory_hostname }}.conf"
      confirm: 5
      check_commit_wait: 3
    register: response
  - name: Print the complete response
    debug:
      var: response
  - name: Confirm the previous commit with a commit check (but no commit)
    juniper_junos_config:
      check: true
      diff: false
      commit: false
    register: response
  - name: Print the complete response
    debug:
      var: response
```

5.7 Return Values

5.8 Notes

Note:

- The NETCONF system service must be enabled on the target Junos device.

5.8.1 Author

- Juniper Networks - Stacy Smith (@stacywsmith)

5.8.2 Status

This module is flagged as **stableinterface** which means that the maintainers for this module guarantee that no backward incompatible interface changes will be made.

CHAPTER 6

juniper_junos_pmtud

Perform path MTU discovery from a Junos device to a destination

New in version 2.0.0.

6.1 Synopsis

- Determine the maximum IP MTU supported along a path from a Junos device to a user-specified destination by performing path MTU discovery (PMTUD) using the ping command. The reported MTU will be between min_test_size and *max_size* where *min_test_size* = (*max_size* - *max_range* + 1). If the actual path MTU is greater than *max_size*, then *max_size* will be reported. If the actual path MTU is less than *min_test_size*, then a failure will be reported.

6.2 Requirements

The following software packages must be installed on hosts that execute this module:

- junos-eznc >= 2.2.0
- Python >= 2.7

6.3 Module-specific Options

The following options may be specified for this module:

6.4 Common Connection-related Options

In addition to the *Module-specific Options*, the following connection-related options are also supported by this module:

6.5 Common Logging-related Options

In addition to the *Module-specific Options*, the following logging-related options are also supported by this module:

6.6 Examples

```
---
- name: Examples of juniper_junos_mtud
  hosts: junos-all
  connection: local
  gather_facts: no
  roles:
    - Juniper.junos

  tasks:
    - name: Perform PMTUD to 192.68.1.1 with default parameters.
      juniper_junos_pmtud:
        dest: "192.68.1.1"

    - name: Perform PMTUD to 192.68.1.1. Register response.
      juniper_junos_pmtud:
        dest: "192.68.1.1"
      register: response
    - name: Print the discovered MTU.
      debug:
        var: response.inet_mtu

    - name: Perform PMTUD to 192.68.1.1. Search all possible MTU values.
      juniper_junos_pmtud:
        dest: "192.68.1.1"
        max_size: 65496
        max_range: 65536
      register: response
```

(continues on next page)

(continued from previous page)

```
  - name: Print the discovered MTU.
    debug:
      var: response.inet_mtu

  - name: Perform PMTUD to 192.68.1.1. Source from ge-0/0/0.0 interface.
    juniper_junos_pmtud:
      dest: "192.68.1.1"
      interface: "ge-0/0/0.0"
    register: response
  - name: Print the discovered MTU.
    debug:
      var: response.inet_mtu

  - name: Perform PMTUD to 192.68.1.1. Source from 192.168.1.2.
    juniper_junos_pmtud:
      dest: "192.68.1.1"
      source: "192.168.1.2"
    register: response
  - name: Print the discovered MTU.
    debug:
      var: response.inet_mtu

  - name: Perform PMTUD to 192.68.1.1. Source from the red routing-instance.
    juniper_junos_pmtud:
      dest: "192.68.1.1"
      routing_instance: "red"
    register: response
  - name: Print the discovered MTU.
    debug:
      var: response.inet_mtu
```

6.7 Return Values

6.8 Notes

Note:

- The NETCONF system service must be enabled on the target Junos device.

6.8.1 Author

- Martin Komon (@mkomon)
- Juniper Networks - Stacy Smith (@stacywsmith)

6.8.2 Status

This module is flagged as **stableinterface** which means that the maintainers for this module guarantee that no backward incompatible interface changes will be made.

CHAPTER 7

juniper_junos_srx_cluster

Add or remove SRX chassis cluster configuration

New in version 2.0.0.

7.1 Synopsis

- Add an SRX chassis cluster configuration and reboot the device. Assuming the device is capable of forming an SRX cluster and has the correct cables connected, this will form an SRX cluster.
- If an SRX chassis cluster is already present, setting *cluster_enable* to `false` will remove the SRX chassis cluster configuration and reboot the device causing the SRX cluster to be broken and the device to return to stand-alone mode.

7.2 Requirements

The following software packages must be installed on hosts that execute this module:

- junos-eznc >= 2.2.0
- Python >= 2.7

7.3 Module-specific Options

The following options may be specified for this module:

7.4 Common Connection-related Options

In addition to the *Module-specific Options*, the following connection-related options are also supported by this module:

7.5 Common Logging-related Options

In addition to the *Module-specific Options*, the following logging-related options are also supported by this module:

7.6 Examples

```
---
- name: Manipulate the SRX cluster configuration of Junos SRX devices
  hosts: junos-all
  connection: local
  gather_facts: no
  roles:
    - Juniper.junos
  tasks:
    - name: Enable an SRX cluster
      juniper_junos_srx_cluster:
        enable: true
        cluster_id: 4
        node_id: 0
      register: response
    - name: Print the response.
      debug:
        var: response.config_lines

    - name: Disable an SRX cluster
      juniper_junos_srx_cluster:
        enable: false
      register: response
    - name: Print the response.
      debug:
        var: response.config_lines
```

7.7 Return Values

7.8 Notes

Note:

- The NETCONF system service must be enabled on the target Junos device.

7.8.1 Author

- Juniper Networks - Stacy Smith (@stacywsmith)

7.8.2 Status

This module is flagged as **stableinterface** which means that the maintainers for this module guarantee that no backward incompatible interface changes will be made.

CHAPTER 8

juniper_junos_table

Retrieve data from a Junos device using a PyEZ table/view

New in version 2.0.0.

8.1 Synopsis

- Retrieve data from a Junos device using PyEZ's operational table/views. This module may be used with the tables/views which are included in the PyEZ distribution or it may be used with user-defined tables/views.

8.2 Requirements

The following software packages must be installed on hosts that execute this module:

- junos-eznc >= 2.2.0
- Python >= 2.7

8.3 Module-specific Options

The following options may be specified for this module:

8.4 Common Connection-related Options

In addition to the *Module-specific Options*, the following connection-related options are also supported by this module:

8.5 Common Logging-related Options

In addition to the *Module-specific Options*, the following logging-related options are also supported by this module:

8.6 Examples

```
---
- name: Retrieve data from a Junos device using a PyEZ table/view.
  hosts: junos-all
  connection: local
  gather_facts: no
  roles:
    - Juniper.junos

  tasks:
    - name: Retrieve LLDP Neighbor Information Using PyEZ-included Table
      juniper_junos_table:
        file: "lldp.yml"
      register: response
    - name: Print response
      debug:
        var: response

    - name: Retrieve routes within 192.68.1/8
      juniper_junos_table:
        file: "routes.yml"
        table: "RouteTable"
        kwargs:
          destination: "192.68.1.0/8"
        response_type: "juniper_items"
      register: response
    - name: Print response
      debug:
        var: response

    - name: Retrieve from custom table in playbook directory
      juniper_junos_table:
```

(continues on next page)

(continued from previous page)

```
      file: "fpc.yaml"
      path: "."
    register: response
  - name: Print response
    debug:
      var: response
```

8.7 Return Values

8.8 Notes

Note:

- This module only works with operational tables/views; it does not work with configuration tables/views.
- The NETCONF system service must be enabled on the target Junos device.

8.8.1 Author

- Jason Edelman (@jedelman8)
- Updated by Juniper Networks - Stacy Smith (@stacywsmith)

8.8.2 Status

This module is flagged as **stableinterface** which means that the maintainers for this module guarantee that no backward incompatible interface changes will be made.

CHAPTER 9

juniper_junos_ping

Execute ping from a Junos device

New in version 2.0.0.

9.1 Synopsis

- Execute the ping command from a Junos device to a specified destination in order to test network reachability from the Junos device .

9.2 Requirements

The following software packages must be installed on hosts that execute this module:

- junos-eznc >= 2.2.0
- Python >= 2.7

9.3 Module-specific Options

The following options may be specified for this module:

9.4 Common Connection-related Options

In addition to the *Module-specific Options*, the following connection-related options are also supported by this module:

9.5 Common Logging-related Options

In addition to the *Module-specific Options*, the following logging-related options are also supported by this module:

9.6 Examples

```
---
- name: Examples of juniper_junos_ping
  hosts: junos-all
  connection: local
  gather_facts: no
  roles:
    - Juniper.junos

  tasks:
    - name: Ping 192.68.1.1 with default parameters. Fails if any packets lost.
      juniper_junos_ping:
        dest: "192.68.1.1"

    - name: Ping 192.68.1.1 Allow 50% packet loss. Register response.
      juniper_junos_ping:
        dest: "192.68.1.1"
        acceptable_percent_loss: 50
      register: response
    - name: Print all keys in the response.
      debug:
        var: response

    - name: Ping 192.68.1.1. Send 20 packets. Register response.
      juniper_junos_ping:
        dest: "192.68.1.1"
        count: 20
      register: response
    - name: Print packet sent from the response.
      debug:
        var: response.packets_sent
```

(continues on next page)

(continued from previous page)

```
  - name: Ping 192.68.1.1. Send 10 packets wihtout rapid. Register response.
    juniper_junos_ping:
      dest: "192.68.1.1"
      count: 10
      rapid: false
    register: response
  - name: Print the average round-trip-time from the response.
    debug:
      var: response.rtt_average

  - name: Ping www.juniper.net with ttl 15. Register response.
    juniper_junos_ping:
      dest: "www.juniper.net"
      ttl: 15
    register: response
  - name: Print the packet_loss percentage from the response.
    debug:
      var: response.packet_loss

  - name: Ping 192.68.1.1 with IP packet size of 1500. Register response.
    juniper_junos_ping:
      dest: "192.68.1.1"
      size: 1472
    register: response
  - name: Print the packets_received from the response.
    debug:
      var: response.packets_received

  - name: Ping 192.68.1.1 with do-not-fragment bit set. Register response.
    juniper_junos_ping:
      dest: "192.68.1.1"
      do_not_fragment: true
    register: response
  - name: Print the maximum round-trip-time from the response.
    debug:
      var: response.rtt_maximum

  - name: Ping 192.68.1.1 with source set to 192.68.1.2. Register response.
    juniper_junos_ping:
      dest: "192.68.1.1"
      source: "192.68.1.2"
    register: response
  - name: Print the source from the response.
    debug:
      var: response.source

  - name: Ping 192.168.1.1 from the red routing-instance.
    juniper_junos_ping:
      dest: "192.168.1.1"
      routing_instance: "red"

  - name: Ping the all-hosts multicast address from the ge-0/0/0.0 interface
    juniper_junos_ping:
      dest: "224.0.0.1"
      interface: "ge-0/0/0.0"
```

9.7 Return Values

9.8 Notes

Note:

- The NETCONF system service must be enabled on the target Junos device.

9.8.1 Author

- Juniper Networks - Stacy Smith (@stacywsmith)

9.8.2 Status

This module is flagged as **stableinterface** which means that the maintainers for this module guarantee that no backward incompatible interface changes will be made.

CHAPTER 10

juniper_junos_system

Initiate operational actions on the Junos system

10.1 Synopsis

- Initiate an operational action (shutdown, reboot, halt or zeroize) on a Junos system. The particular action to execute is defined by the mandatory *action* option.

10.2 Requirements

The following software packages must be installed on hosts that execute this module:

- junos-eznc >= 2.2.0

- Python >= 2.7

10.3 Module-specific Options

The following options may be specified for this module:

10.4 Common Connection-related Options

In addition to the *Module-specific Options*, the following connection-related options are also supported by this module:

10.5 Common Logging-related Options

In addition to the *Module-specific Options*, the following logging-related options are also supported by this module:

10.6 Examples

```
---
- name: Examples of juniper_junos_system
  hosts: junos-all
  connection: local
  gather_facts: no
  roles:
    - Juniper.junos

  tasks:
    - name: Reboot all REs of the device
      juniper_junos_system:
        action: "reboot"

    - name: Power off the other RE of the device.
      juniper_junos_system:
        action: "shutdown"
        othe_re: True

    - name: Reboot this RE at 8pm today.
      juniper_junos_system:
        action: "reboot"
        all_re: False
        at: "20:00"

    - name: Halt the system on 25 January 2018 at 4pm.
      juniper_junos_system:
        action: "halt"
        at: "1801251600"

    - name: Reboot the system in 30 minutes.
      juniper_junos_system:
        action: "reboot"
        in_min: 30
```

(continues on next page)

(continued from previous page)

```
- name: Reboot the system in 30 minutes.
  juniper_junos_system:
    action: "reboot"
    at: "+30m"

- name: Zeroize the local RE only.
  juniper_junos_system:
    action: "zeroize"
    all_re: False

- name: Zeroize all REs and overwrite medea.
  juniper_junos_system:
    action: "zeroize"
    media: True
```

10.7 Return Values

10.8 Notes

Note:

- This module only **INITIATES** the action. It does **NOT** wait for the action to complete.
- Some Junos devices are effected by a Junos defect which causes this Ansible module to hang indefinitely when connected to the Junos device via the console. This problem is not seen when connecting to the Junos device using the normal NETCONF over SSH transport connection. Therefore, it is recommended to use this module only with a NETCONF over SSH transport connection. However, this module does still permit connecting to Junos devices via the console port and this functionality may still be used for Junos devices running Junos versions less than 15.1.
- The NETCONF system service must be enabled on the target Junos device.

10.8.1 Author

- Juniper Networks - Stacy Smith (@stacywsmith)

10.8.2 Status

This module is flagged as **stableinterface** which means that the maintainers for this module guarantee that no backward incompatible interface changes will be made.

CHAPTER 11

juniper_junos_rpc

Execute one or more NETCONF RPCs on a Junos device

New in version 2.0.0.

11.1 Synopsis

- Execute one or more NETCONF RPCs on a Junos device.
- Use the `| display xml rpc` modifier to determine the equivalent RPC name for a Junos CLI command. For example, `show version | display xml rpc` reveals the equivalent RPC name is `get-software-information`.

11.2 Requirements

The following software packages must be installed on hosts that execute this module:

- junos-eznc >= 2.2.0
- Python >= 2.7

11.3 Module-specific Options

The following options may be specified for this module:

11.4 Common Connection-related Options

In addition to the *Module-specific Options*, the following connection-related options are also supported by this module:

11.5 Common Logging-related Options

In addition to the *Module-specific Options*, the following logging-related options are also supported by this module:

11.6 Examples

```
---
- name: Examples of juniper_junos_rpc
  hosts: junos-all
  connection: local
  gather_facts: no
  roles:
    - Juniper.junos

  tasks:
    - name: Execute single get-software-information RPC.
      juniper_junos_rpc:
        rpcs: "get-software-information"
      register: response
    - name: Print the RPC's output as a single multi-line string.
      debug:
        var: response.stdout

###### OLD EXAMPLES ##########
- junos_rpc:
  host={{ inventory_hostname }}
  rpc=get-interface-information
  dest=get_interface_information.conf
  register=junos

- junos_rpc:
  host={{ inventory_hostname }}
  rpc=get-interface-information
```

(continues on next page)

(continued from previous page)

```
  kwargs="interface_name=em0"
  format=xml/text/json
  dest=get_interface_information.conf
  register=junos

# Example to fetch device configuration
- name: Get Device Configuration
  junos_rpc:
    host={{ inventory_hostname }}
    rpc=get-config
    dest=get_config.conf

# Fetch configuration over console server connection using PyEZ >= 2.0
- name: Get Device Configuration
  junos_rpc:
    host={{ inventory_hostname }}
    port=7005
    mode='telnet'
    rpc=get-config
    dest=get_config.conf

# Example to fetch device configuration
- name: Get Device Configuration for interface
  junos_rpc:
    host={{ inventory_hostname }}
    rpc=get-config
    filter_xml="<configuration><interfaces/></configuration>"
    dest=get_config.conf
  register: junos

# Example to fetch configuration in json for >=14.2
# and use it with rpc_reply
- name: Get Device Configuration
  hosts: all
  roles:
    - Juniper.junos
  connection: local
  gather_facts: no
  tasks:
    - name: Get interface information
      junos_rpc:
        host: "{{ inventory_hostname }}"
        rpc: get-interface-information
        kwargs:
          interface_name: em0
          media: True
        format: json
        dest: get_interface_information.conf
      register: junos

    - name: Print configuration
      debug: msg="{{ junos.rpc_reply }}"
###### OLD EXAMPLES ##########
```

11.7 Return Values

11.8 Notes

Note:

- The NETCONF system service must be enabled on the target Junos device.

11.8.1 Author

- Juniper Networks - Stacy Smith (@stacywsmith)

11.8.2 Status

This module is flagged as **stableinterface** which means that the maintainers for this module guarantee that no backward incompatible interface changes will be made.