Network Resource Modules

Ansible network resource modules simplify and standardize how you manage different network devices. Network devices separate configuration into sections (such as interfaces and VLANs) that apply to a network service. Ansible network resource modules take advantage of this to allow you to configure subsections or *resources* within the network device configuration. Network resource modules provide a consistent experience across different network devices.

Network resource module states

You use the network resource modules by assigning a state to what you want the module to do. The resource modules support the following states:

merged
: Ansible merges the on-device configuration with the provided configuration in the task.

replaced
: Ansible replaces the on-device configuration subsection with the provided configuration subsection in the task.

overridden
: Ansible overrides the on-device configuration for the resource with the provided configuration in the task. Use caution with this state as you could remove your access to the device (for example, by overriding the management interface configuration).

deleted
: Ansible deletes the on-device configuration subsection and restores any default settings.

gathered
: Ansible displays the resource details gathered from the network device and accessed with the `gathered` key in the result.

rendered
: Ansible renders the provided configuration in the task in the device-native format (for example, Cisco IOS CLI). Ansible returns this rendered configuration in the `rendered` key in the result. Note this state does not communicate with the network device and can be used offline.

parsed
: Ansible parses the configuration from the `running_config` option into Ansible structured data in the `parsed` key in the result. Note this does not gather the configuration from the network device so this state can be used offline.

Using network resource modules

This example configures the L3 interface resource on a Cisco IOS device, based on different state settings.

```
- name: configure l3 interface
  cisco.ios.ios_l3_interfaces:
    config: "{{ config }}"
    state: <state>
```

The following table shows an example of how an initial resource configuration changes with this task for different states.

Resource starting configuration	task-provided configuration (YAML)	Final resource configuration on device
`interface loopback100` `ip address 10.10.1.100 255.255.255.0` `ipv6 address FC00:100/64`	`config:` `- ipv6:` `- address: fc00::100/64` `- address: fc00::101/64` `name: loopback100`	*merged* `interface loopback100` `ip address 10.10.1.100 255.255.255` `ipv6 address FC00:100/64` `ipv6 address FC00:101/64`
		replaced `interface loopback100` `no ip address` `ipv6 address FC00:100/64` `ipv6 address FC00:101/64`
		overridden Incorrect use case. This would remove all interfaces from the device (including the mgmt interface) except the configured loopback100
		deleted `interface loopback100` `no ip address`

Network resource modules return the following details:

- The *before* state - the existing resource configuration before the task was executed.
- The *after* state - the new resource configuration that exists on the network device after the task was executed.
- Commands - any commands configured on the device.

```
ok: [nxos101] =>
  result:
    after:
      contact: IT Support
      location: Room E, Building 6, Seattle, WA 98134
      users:
      - algorithm: md5
        group: network-admin
        localized_key: true
        password: '0x73fd9a2cc8c53ed3dd4ed8f4ff157e69'
        privacy_password: '0x73fd9a2cc8c53ed3dd4ed8f4ff157e69'
        username: admin
    before:
      contact: IT Support
      location: Room E, Building 5, Seattle HQ
      users:
      - algorithm: md5
        group: network-admin
        localized_key: true
        password: '0x73fd9a2cc8c53ed3dd4ed8f4ff157e69'
        privacy_password: '0x73fd9a2cc8c53ed3dd4ed8f4ff157e69'
        username: admin
    changed: true
    commands:
    - snmp-server location Room E, Building 6, Seattle, WA 98134
    failed: false
```

Example: Verifying the network device configuration has not changed

The following playbook uses the :ref:`arista.eos.eos_l3_interfaces <ansible_collections.arista.eos.eos_l3_interfaces_module>` module to gather a subset of the network device configuration (Layer 3 interfaces only) and verifies the information is accurate and has not changed. This playbook passes the results of :ref:`arista.eos.eos_facts <ansible_collections.arista.eos.eos_facts_module>` directly to the `arista.eos.eos_l3_interfaces` module.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\network\user_guide\[ansible-devel][docs][docsite][rst][network][user_guide]network_resource_modules.rst`, line 124); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\network\user_guide\[ansible-devel][docs][docsite][rst][network][user_guide]network_resource_modules.rst`, line 124); *backlink*

Unknown interpreted text role "ref".

```
- name: Example of facts being pushed right back to device.
  hosts: arista
  gather_facts: false
  tasks:
    - name: grab arista eos facts
      arista.eos.eos_facts:
        gather_subset: min
        gather_network_resources: l3_interfaces

- name: Ensure that the IP address information is accurate.
  arista.eos.eos_l3_interfaces:
    config: "{{ ansible_network_resources['l3_interfaces'] }}"
    register: result

- name: Ensure config did not change.
  assert:
    that: not result.changed
```

Example: Acquiring and updating VLANs on a network device

This example shows how you can use resource modules to:

1. Retrieve the current configuration on a network device.
2. Save that configuration locally.
3. Update that configuration and apply it to the network device.

This example uses the `cisco.ios.ios_vlans` resource module to retrieve and update the VLANs on an IOS device.

1. Retrieve the current IOS VLAN configuration:

```
- name: Gather VLAN information as structured data
  cisco.ios.ios_facts:
     gather_subset:
      - '!all'
      - '!min'
     gather_network_resources:
     - 'vlans'
```

2. Store the VLAN configuration locally:

```
- name: Store VLAN facts to host_vars
  copy:
    content: "{{ ansible_network_resources | to_nice_yaml }}"
    dest: "{{ playbook_dir }}/host_vars/{{ inventory_hostname }}"
```

3. Modify the stored file to update the VLAN configuration locally.
4. Merge the updated VLAN configuration with the existing configuration on the device:

```
- name: Make VLAN config changes by updating stored facts on the controller.
  cisco.ios.ios_vlans:
    config: "{{ vlans }}"
    state: merged
  tags: update_config
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\network\user_guide\[ansible-devel][docs][docsite][rst][network][user_guide]network_resource_modules.rst`, line 191)

Unknown directive type "seealso".

```
.. seealso::

  `Network Features in Ansible 2.9 <https://www.ansible.com/blog/network-features-coming-soon-in-ansible-eng
    A introductory blog post on network resource modules.
  `Deep Dive into Network Resource Modules <https://www.ansible.com/deep-dive-into-ansible-network-resource-
    A deeper dive presentation into network resource modules.
```