Controlling how Ansible behaves: precedence rules

To give you maximum flexibility in managing your environments, Ansible offers many ways to control how Ansible behaves: how it connects to managed nodes, how it works once it has connected. If you use Ansible to manage a large number of servers, network devices, and cloud resources, you may define Ansible behavior in several different places and pass that information to Ansible in several different ways. This flexibility is convenient, but it can backfire if you do not understand the precedence rules.

These precedence rules apply to any setting that can be defined in multiple ways (by configuration settings, command-line options, playbook keywords, variables).

Precedence categories

Ansible offers four sources for controlling its behavior. In order of precedence from lowest (most easily overridden) to highest (overrides all others), the categories are:

- Configuration settings
- Command-line options
- Playbook keywords
- Variables

Each category overrides any information from all lower-precedence categories. For example, a playbook keyword will override any configuration setting.

Within each precedence category, specific rules apply. However, generally speaking, 'last defined' wins and overrides any previous definitions.

Configuration settings

:ref:`Configuration settings<ansible_configuration_settings>` include both values from the `ansible.cfg` file and environment variables. Within this category, values set in configuration files have lower precedence. Ansible uses the first `ansible.cfg` file it finds, ignoring all others. Ansible searches for `ansible.cfg` in these locations in order:

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]general_precedence.rst`, line 32); *backlink***
>
> Unknown interpreted text role "ref".

- `ANSIBLE_CONFIG` (environment variable if set)
- `ansible.cfg` (in the current directory)
- `~/.ansible.cfg` (in the home directory)
- `/etc/ansible/ansible.cfg`

Environment variables have a higher precedence than entries in `ansible.cfg`. If you have environment variables set on your control node, they override the settings in whichever `ansible.cfg` file Ansible loads. The value of any given environment variable follows normal shell precedence: the last value defined overwrites previous values.

Command-line options

Any command-line option will override any configuration setting.

When you type something directly at the command line, you may feel that your hand-crafted values should override all others, but Ansible does not work that way. Command-line options have low precedence - they override configuration only. They do not override playbook keywords, variables from inventory or variables from playbooks.

You can override all other settings from all other sources in all other precedence categories at the command line by :ref:`general_precedence_extra_vars`, but that is not a command-line option, it is a way of passing a :ref:`variable<general_precedence_variables>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]general_precedence.rst, line 48); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]general_precedence.rst, line 48); *backlink*

Unknown interpreted text role "ref".

At the command line, if you pass multiple values for a parameter that accepts only a single value, the last defined value wins. For example, this :ref:`ad hoc task<intro_adhoc>` will connect as `carol`, not as `mike`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]general_precedence.rst, line 50); *backlink*

Unknown interpreted text role "ref".

```
ansible -u mike -m ping myhost -u carol
```

Some parameters allow multiple values. In this case, Ansible will append all values from the hosts listed in inventory files inventory1 and inventory2:

```
ansible -i /path/inventory1 -i /path/inventory2 -m ping all
```

The help for each :ref:`command-line tool<command_line_tools>` lists available options for that tool.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]general_precedence.rst, line 58); *backlink*

Unknown interpreted text role "ref".

Playbook keywords

Any :ref:`playbook keyword<playbook_keywords>` will override any command-line option and any configuration setting.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]general_precedence.rst, line 63); *backlink*

Unknown interpreted text role "ref".

Within playbook keywords, precedence flows with the playbook itself; the more specific wins against the more general:

- play (most general)
- blocks/includes/imports/roles (optional and can contain tasks and each other)
- tasks (most specific)

A simple example:

```
- hosts: all
  connection: ssh
  tasks:
    - name: This task uses ssh.
      ping:

    - name: This task uses paramiko.
      connection: paramiko
      ping:
```

In this example, the `connection` keyword is set to `ssh` at the play level. The first task inherits that value, and connects using `ssh`. The second task inherits that value, overrides it, and connects using `paramiko`. The same logic applies to blocks and roles as well. All tasks, blocks, and roles within a play inherit play-level keywords; any task, block, or role can override any keyword by defining a different value for that keyword within the task, block, or role.

Remember that these are KEYWORDS, not variables. Both playbooks and variable files are defined in YAML but they have different significance. Playbooks are the command or 'state description' structure for Ansible, variables are data we use to help make

playbooks more dynamic.

Variables

Any variable will override any playbook keyword, any command-line option, and any configuration setting.

Variables that have equivalent playbook keywords, command-line options, and configuration settings are known as :ref:`connection_variables`. Originally designed for connection parameters, this category has expanded to include other core variables like the temporary directory and the python interpreter.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]general_precedence.rst`, line 96); *backlink*
>
> Unknown interpreted text role "ref".

Connection variables, like all variables, can be set in multiple ways and places. You can define variables for hosts and groups in :ref:`inventory<intro_inventory>`. You can define variables for tasks and plays in `vars:` blocks in :ref:`playbooks<about_playbooks>`. However, they are still variables - they are data, not keywords or configuration settings. Variables that override playbook keywords, command-line options, and configuration settings follow the same rules of :ref:`variable precedence <ansible_variable_precedence>` as any other variables.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]general_precedence.rst`, line 98); *backlink*
>
> Unknown interpreted text role "ref".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]general_precedence.rst`, line 98); *backlink*
>
> Unknown interpreted text role "ref".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]general_precedence.rst`, line 98); *backlink*
>
> Unknown interpreted text role "ref".

When set in a playbook, variables follow the same inheritance rules as playbook keywords. You can set a value for the play, then override it in a task, block, or role:

```
- hosts: cloud
  gather_facts: false
  become: yes
  vars:
    ansible_become_user: admin
  tasks:
    - name: This task uses admin as the become user.
      dnf:
        name: some-service
        state: latest
    - block:
        - name: This task uses service-admin as the become user.
          # a task to configure the new service
        - name: This task also uses service-admin as the become user, defined in the block.
          # second task to configure the service
      vars:
        ansible_become_user: service-admin
    - name: This task (outside of the block) uses admin as the become user again.
      service:
        name: some-service
        state: restarted
```

Variable scope: how long is a value available?

Variable values set in a playbook exist only within the playbook object that defines them. These 'playbook object scope' variables are not available to subsequent objects, including other plays.

Variable values associated directly with a host or group, including variables defined in inventory, by vars plugins, or using modules like :ref:`set_fact<set_fact_module>` and :ref:`include_vars<include_vars_module>`, are available to all plays. These 'host scope'

variables are also available via the `hostvars[]` dictionary.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]general_precedence.rst, line 129); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\reference_appendices\[ansible-devel][docs][docsite][rst][reference_appendices]general_precedence.rst, line 129); *backlink*

Unknown interpreted text role "ref".

Using `-e` extra variables at the command line

To override all other settings in all other categories, you can use extra variables: `--extra-vars` or `-e` at the command line. Values passed with `-e` are variables, not command-line options, and they will override configuration settings, command-line options, and playbook keywords as well as variables set elsewhere. For example, this task will connect as `brian` not as `carol`:

```
ansible -u carol -e 'ansible_user=brian' -a whoami all
```

You must specify both the variable name and the value with `--extra-vars`.