ansible.builtin.systemd module – Manage systemd units

Note

This module is part of `ansible-core` and included in all Ansible installations. In most cases, you can use the short module name `systemd` even without specifying the `collections:` keyword. However, we recommend you use the FQCN for easy linking to the module documentation and to avoid conflicting with other collections that may have the same module name.

Synopsis

- Controls systemd units (services, timers, and so on) on remote hosts.

Requirements

The below requirements are needed on the host that executes this module.

- A system managed by systemd.

Parameters

daemon_reexec
aliases: daemon-reexec
boolean
added in Ansible 2.8

Run daemon_reexec command before doing any other operations, the systemd manager will serialize the manager state.

Choices:

- **`false`** ← (default)
- `true`

daemon_reload
aliases: daemon-reload
boolean

Run daemon-reload before doing any other operations, to make sure systemd has read any changes.

When set to `true`, runs daemon-reload even if the module does not start or stop anything.

Choices:

- **`false`** ← (default)
- `true`

enabled
boolean

Whether the unit should start on boot. **At least one of state and enabled are required.**

Choices:

- `false`
- `true`

force
boolean

Whether to override existing symlinks.

Choices:

- `false`
- `true`

masked
boolean

Whether the unit should be masked or not, a masked unit is impossible to start.

Choices:

- `false`
- `true`

name
aliases: service, unit
string

Name of the unit. This parameter takes the name of exactly one unit to work with.

When no extension is given, it is implied to a `.service` as systemd.

When using in a chroot environment you always need to specify the name of the unit with the extension. For example, `crond.service`.

no_block
boolean

Do not synchronously wait for the requested operation to finish. Enqueued job will continue without Ansible blocking on its completion.

Choices:

- **`false`** ← (default)
- `true`

scope
string
added in Ansible 2.7

Run systemctl within a given service manager scope, either as the default system scope `system`, the current user's scope `user`, or the scope of all users `global`.

For systemd to work with 'user', the executing user must have its own instance of dbus started and accessible (systemd requirement).

The user dbus process is normally started during normal login, but not during the run of Ansible tasks. Otherwise you will probably get a 'Failed to connect to bus: no such file or directory' error.

The user must have access, normally given via setting the `XDG_RUNTIME_DIR` variable, see example below.

Choices:

- **`"system"`** ← (default)
- `"user"`
- `"global"`

state
string

`started` / `stopped` are idempotent actions that will not run commands unless necessary. `restarted` will always bounce the unit. `reloaded` will always reload.

Choices:

- `"reloaded"`
- `"restarted"`
- `"started"`
- `"stopped"`

Attributes

check_mode

Support: full

Can run in check_mode and return changed status prediction without modifying target

diff_mode

Support: none

Will return details on what has changed (or possibly needs changing in check_mode), when in diff mode

platform

Platform: posix

Target OS/families that can be operated against

Notes

Note

- Since 2.4, one of the following options is required `state`, `enabled`, `masked`, `daemon_reload`, (`daemon_reexec` since 2.8), and all except `daemon_reload` and (`daemon_reexec` since 2.8) also require `name`.
- Before 2.4 you always required `name`.
- Globs are not supported in name, i.e `postgres*.service`.
- The service names might vary by specific OS/distribution
- The order of execution when having multiple properties is to first enable/disable, then mask/unmask and then deal with service state. It has been reported that systemctl can behave differently depending on the order of operations if you do the same manually.

Examples

```
- name: Make sure a service unit is running
  ansible.builtin.systemd:
    state: started
    name: httpd

- name: Stop service cron on debian, if running
  ansible.builtin.systemd:
    name: cron
    state: stopped

- name: Restart service cron on centos, in all cases, also issue daemon-reload to pick up config changes
  ansible.builtin.systemd:
    state: restarted
    daemon_reload: true
    name: crond

- name: Reload service httpd, in all cases
  ansible.builtin.systemd:
    name: httpd.service
    state: reloaded

- name: Enable service httpd and ensure it is not masked
  ansible.builtin.systemd:
    name: httpd
    enabled: true
    masked: no

- name: Enable a timer unit for dnf-automatic
  ansible.builtin.systemd:
    name: dnf-automatic.timer
    state: started
    enabled: true

- name: Just force systemd to reread configs (2.4 and above)
  ansible.builtin.systemd:
    daemon_reload: true

- name: Just force systemd to re-execute itself (2.8 and above)
  ansible.builtin.systemd:
    daemon_reexec: true

- name: Run a user service when XDG_RUNTIME_DIR is not set on remote login
  ansible.builtin.systemd:
    name: myservice
    state: started
    scope: user
  environment:
    XDG_RUNTIME_DIR: "/run/user/{{ myuid }}"
```

Return Values

Common return values are documented here (../../../reference_appendices/common_return_values.html#common-return-values), the following are the fields unique to this module:

status
complex

A dictionary with the key=value pairs returned from `systemctl show`.

Returned: success

Sample:

```
{"ActiveEnterTimestamp": "Sun 2016-05-15 18:28:49 EDT", "ActiveEnterTimestampMonotonic": "8135942", "ActiveExitTimestampMonotonic": "0", "ActiveState": "active", "After": "auditd.service systemd-user-sessions.service time-sync.target systemd-journald.socket basic.target system.slice", "AllowIsolate": "no", "Before": "shutdown.target multi-user.target", "BlockIOAccounting": "no", "BlockIOWeight": "1000", "CPUAccounting": "no", "CPUSchedulingPolicy": "0", "CPUSchedulingPriority": "0", "CPUSchedulingResetOnFork": "no", "CPUShares": "1024", "CanIsolate": "no", "CanReload": "yes", "CanStart": "yes", "CanStop": "yes", "CapabilityBoundingSet": "18446744073709551615", "ConditionResult": "yes", "ConditionTimestamp": "Sun 2016-05-15 18:28:49 EDT", "ConditionTimestampMonotonic": "7902742", "Conflicts": "shutdown.target", "ControlGroup": "/system.slice/crond.service", "ControlPID": "0", "DefaultDependencies": "yes", "Delegate": "no", "Description": "Command Scheduler", "DevicePolicy": "auto", "EnvironmentFile": "/etc/sysconfig/crond (ignore_errors=no)", "ExecMainCode": "0", "ExecMainExitTimestampMonotonic": "0", "ExecMainPID": "595", "ExecMainStartTimestamp": "Sun 2016-05-15 18:28:49 EDT", "ExecMainStartTimestampMonotonic": "8134990", "ExecMainStatus": "0", "ExecReload": "{ path=/bin/kill ; argv[]=/bin/kill -HUP $MAINPID ; ignore_errors=no ; start_time=[n/a] ; stop_time=[n/a] ; pid=0 ; code=(null) ; status=0/0 }", "ExecStart": "{ path=/usr/sbin/crond ; argv[]=/usr/sbin/crond -n $CRONDARGS ; ignore_errors=no ; start_time=[n/a] ; stop_time=[n/a] ; pid=0 ; code=(null) ; status=0/0 }", "FragmentPath": "/usr/lib/systemd/system/crond.service", "GuessMainPID": "yes", "IOScheduling": "0", "Id": "crond.service", "IgnoreOnIsolate": "no", "IgnoreOnSnapshot": "no", "IgnoreSIGPIPE": "yes", "InactiveEnterTimestampMonotonic": "0", "InactiveExitTimestamp": "Sun 2016-05-15 18:28:49 EDT", "InactiveExitTimestampMonotonic": "8135942", "JobTimeoutUSec": "0", "KillMode": "process", "KillSignal": "15", "LimitAS": "18446744073709551615", "LimitCORE": "18446744073709551615", "LimitCPU": "18446744073709551615", "LimitDATA": "18446744073709551615", "LimitFSIZE": "18446744073709551615", "LimitLOCKS": "18446744073709551615", "LimitMEMLOCK": "65536", "LimitMSGQUEUE": "819200", "LimitNICE": "0", "LimitNOFILE": "4096", "LimitNPROC": "3902", "LimitRSS": "18446744073709551615", "LimitRTPRIO": "0", "LimitRTTIME": "18446744073709551615", "LimitSIGPENDING": "3902", "LimitSTACK": "18446744073709551615", "LoadState": "loaded", "MainPID": "595", "MemoryAccounting": "no", "MemoryLimit": "18446744073709551615", "MountFlags": "0", "Names": "crond.service", "NeedDaemonReload": "no", "Nice": "0", "NoNewPrivileges": "no", "NonBlocking": "no", "NotifyAccess": "none", "OOMScoreAdjust": "0", "OnFailureIsolate": "no", "PermissionsStartOnly": "no", "PrivateNetwork": "no", "PrivateTmp": "no", "RefuseManualStart": "no", "RefuseManualStop": "no", "RemainAfterExit": "no", "Requires": "basic.target", "Restart": "no", "RestartUSec": "100ms", "Result": "success", "RootDirectoryStartOnly": "no", "SameProcessGroup": "no", "SecureBits": "0", "SendSIGHUP": "no", "SendSIGKILL": "yes", "Slice": "system.slice", "StandardError": "inherit", "StandardInput": "null", "StandardOutput": "journal", "StartLimitAction": "none", "StartLimitBurst": "5", "StartLimitInterval": "10000000", "StatusErrno": "0", "StopWhenUnneeded": "no", "SubState": "running", "SyslogLevelPrefix": "yes", "SyslogPriority": "30", "TTYReset": "no", "TTYVHangup": "no", "TTYVTDisallocate": "no", "TimeoutStartUSec": "1min 30s", "TimeoutStopUSec": "1min 30s", "TimerSlackNSec": "50000", "Transient": "no", "Type": "simple", "UMask": "0022", "UnitFileState": "enabled", "WantedBy": "multi-user.target", "Wants": "system.slice", "WatchdogTimestampMonotonic": "0", "WatchdogUSec": "0"}
```

Authors

- Ansible Core Team

Collection links

Issue Tracker (https://github.com/ansible/ansible/issues)

Repository (Sources) (https://github.com/ansible/ansible)

Communication (./#communication-for-ansible-builtin)