Table of Contents

Vendor State Actions

Tasker

When executing jobs to interact with the network, the job is first added to the scheduler which then launches the job at the appropriate time. This job is executed using the ***Tasker***.

Every job has a mandatory component called the ***Scenario*** which consists of step-by-step action commands that perform the desired action on or for a device. Many commands interact with nodes (devices) to retrieve information or alter the configuration. Other commands deal with program execution, yce-database or yce-integrations. These action commands are standardized to perform generic tasks like 'generate commands from a template' and 'execute commands on the CLI'. The scenario program execution commands allow for error checking and conditional actions.

> The action commands with their options are defined in Scenario Command Details.

Since these action commands need to interact with the various devices in specific ways, unique to the vendor, a method is needed to define how these actions need to be executed.

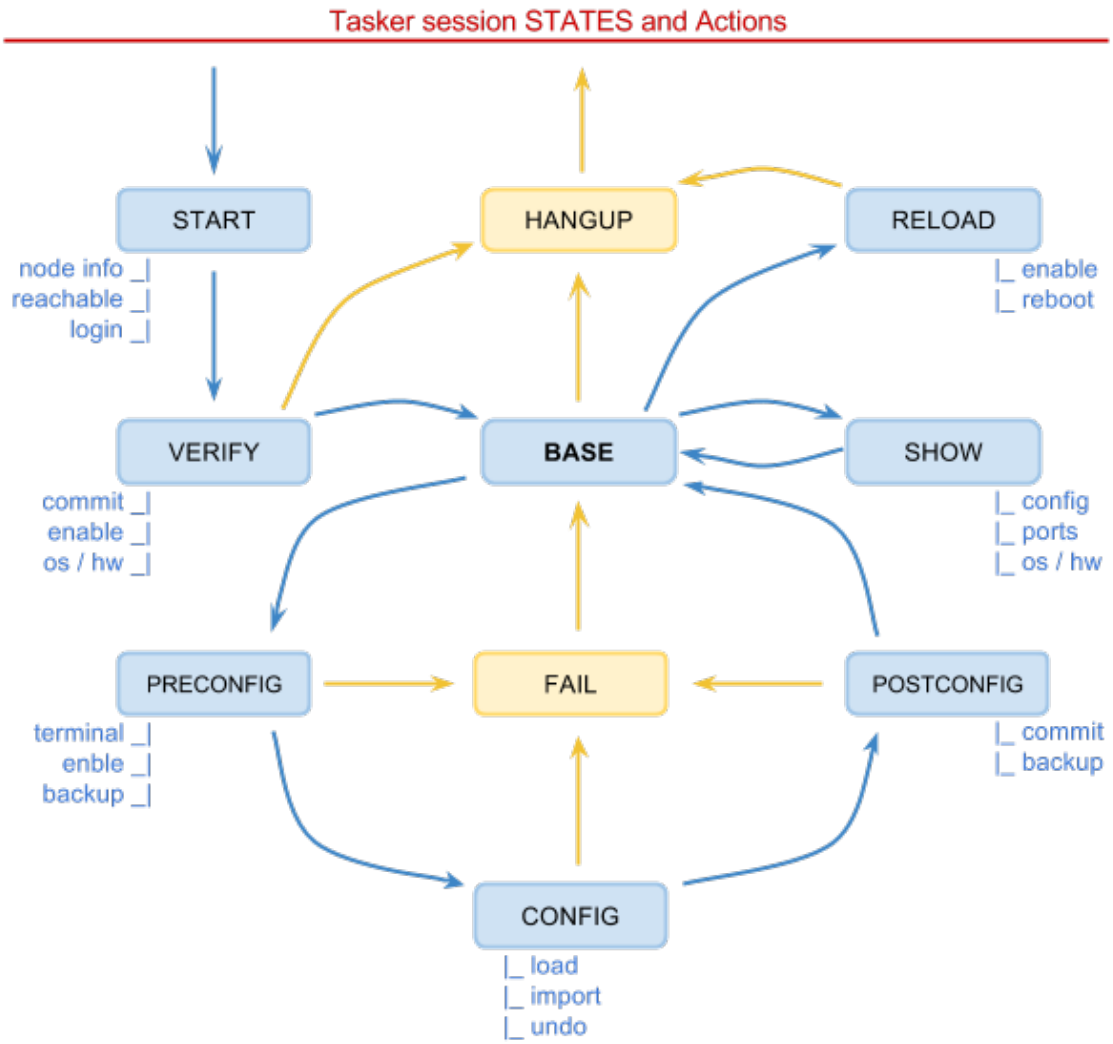
To this end, the Tasker uses a session-state-machine with a fixed set of states that define the possible sequences that session state changes can take. Per state a number of smaller actions are then executed.

Tasker Session-states

For each session, the Tasker will maintain a state-machine. After login and verification, the '**Base**' state is assumed from where all scenario commands will be executed and return to. The '**Config**' state will be used to modify the device configuration in the fashion dictated by the scenario command, but will always be preceded by the '**PreConfig**' state and followed by the '**PostConfig**' state.

Scenario commands not modifying the configuration will be handled in the 'Show' state. The '**Reload**' state is an exception since it should result in a hangup of the session.

When encountering a node-command, the Tasker will establish the indicated node-session or resume an existing session. After each command the session must be returned to the '**Base**' state in order to resume the session.



Multiple Sessions

These states reflect the state of a 'Session'. A node normally is accessed over its 'management' ip-address, but it could also be accessed over its serial 'console' interface or 'out-of-band' management interface. Each of these sessions have their own session state and can easily be active ate the same time.

The Tasker maintains multiple sessions for multiple nodes when required. This is all transparent to the user, his task is limited to defining what action commands need to be executed on which nodes. Specifying the session interface type is an option that is usually ignored.

Sessions once established remain open and active for the duration of the scenario execution, allowing the scenario direct access to multiple nodes to make changes and do verifications.

While a session is in the “base” state waiting for possible command resuming it, it must be prevented from timing-out a and disconnecting the session. A periodic (say 3 min) non-operation command can keep such a session alive and determine if the session is gone nevertheless. For Cisco such an 'idle command' could be “show privilege”.

State_actions table

When assuming a state, the appropriate “Actions” will be executed corresponding to the scenario command. Since these Actions and their order will differ per Vendor-type and may not be available using all of the session-types, a complex set hardcoded routines may ensue.

To prevent this situation and to allow flexible customization and debugging, the logic of state-actions versus vendor and session-type are controlled using a database table. This is realized using the '**YCE.State_actions**' table.

Some sample records of this table:

Id	Vendor_type	Command	State	Action	Seq	Session_types
1001	Cisco_ios		start	node_info	1	all
1003	Cisco_ios		start	reachable	2	mgmt
1005	Cisco_ios		start	login	3	mgmt
1007	Cisco_ios		verify	enable	1	all
1009	Cisco_ios		verify	backup	2	mgmt
1011	Cisco_ios		verify	check_os	3	all
1013	Cisco_ios		base	disable	1	all
1015	Cisco_ios		base	keep_alive	2	mgmt console
1017	Cisco_ios		preconfig	no_alive	0	mgmt console
1019	Cisco_ios	import_cfg	preconfig	terminal	1	mgmt console
1021	Cisco_ios	import_cfg	preconfig	enable	2	all
1023	Cisco_ios	import_cfg	preconfig	backup	3	mgmt
1025	Cisco_ios	import_cfg	config	import	1	all
1027	Cisco_ios	import_cfg	postconfig	backup	1	mgmt
1029	Cisco_ios	import_cfg	base	special	0	mgmt console

The table uses general entries without a 'Command' for 'Start', 'Verify' and 'Base' states. These blank command entries are used for all commands.

This modification also permits to include additional, command-specific, actions in another state: add a command-specific action in a state that otherwise only include blank commands. Or include a default action when entering a state. This usage is demonstrated by the keep_alive and no_alive actions where a periodic hello is sent while in the base state but should cease during configs.