Policies, rules & conditions

Policies

A policy is coupled to a number of node groups. Whenever a node is checked for compliance, we therefore first check its node groups, and then all policies coupled to those node groups, and all those policies get validated on this node. A policy has the following attributes:

- **Policy_id:** The policy's id, a number
- **Policy_name:** The policy's name, this can be any string up to 100 characters
- **Policy_description:** The policy's description, explaining what the policy does, this can be any string
- **Policy_status:** A bitmask.
  - Bit 1: Is the policy enabled? If not, then the policy won't be run
  - Bit 2: Will the policy be run on config change?
- **Signal_trigger:** A bitmask. Determines when a signal will get triggered:
  - Bit 1: When a node switches from compliant to non-compliant
  - Bit 2: When a node switches from non-compliant to compliant
  - Bit 4: When a node switches from non-compliant to non-compliant
  - Bit 8: When a node switches from compliant to compliant
- **Signal_type:** A bitmask. Determines what kind of signal should be sent
  - Bit 1: A trap message
  - Bit 2: A syslog message
  - Bit 4: An email
  - Bit 8: A JSON call

Rules

A policy contains one or more rules. A rule applies to a specific vendor type and either the entire config of a node, or a specific block inside the node's config. Rules contain these attributes

- **Policy_id:** The rule's Policy id
- **Rule_id:** The rule's id
- **Rule_name:** The rule's name
- **Rule_description:** The rule's description
- **Rule_type:** A number for the rule's type:
  - 0: Configuration: The rule applies to a part of the config, defined by the Rule_start and Rule_end attributes
  - 1: Command: The rule applies to the results of a command, performed on the node. This is not yet supported.
  - 3: Multiconfig: The rule compares the config to the configs of the other nodes in its node group.
- **Rule_scope:** A boolean to show in which way configs will be parsed to search for Rule_start and Rule_end
  - 0: Search for the lines between and including the lines that match Rule_start and Rule_end
  - 1: Split the config up into blocks and match the blocks starting with Rule_start. Also works hierarchical and sub-blocks can be found by the first lines of their parents.

- **Enabled:** If the rule is enabled
  - 0: The rule is not enabled and will be ignored when evaluating compliance
  - 1: The rule is enabled. Default.
- **Rule_severity:** The rule's severity. Defaults to 0 (Low), 1 (Medium), 2 (Serious), 3 (High), can be edited in the NCCM Lookup under the Rule_severity.
- **Vendor_type:** The rule's vendor type. This rule will only be run on nodes with this vendor type
- **Rule_start:** The line in the node's config from where the rule should start checking its compliance. Can contain regex, if empty the entire config is taken.
- **Rule_end:** The line in the node's config from where the rule should stop checking its compliance. If empty it takes the end of the block, started at Rule_start.
- **Command:** A command rule's command. Currently not yet supported.
- **Template:** Currently not used.

Conditions

A rule can have a number of conditions. There are two types of conditions: validation and logic conditions. A rule's conditions are in sequence. Together they form a condition logic that needs to comply with the rule's config block in its entirety for the rule to be compliant. A condition has the following attributes:

- **Rule_id:** The condition's Rule id
- **Condition_id:** The condition's id
- **Condition_seq:** The sequence in which the condition is checked when its rule is run for compliance
- **Condition_name:** The condition's name. NOTE that this is not like the names for policies and rules. In the case of a logic condition this is the condition's statement, else this is a word, preferably a single letter.
- **Condition_type:** A string containing the condition's type. This type can differ, depending on the rule and the condition
  - A logic condition can have type If, Then, Else, And, Or, (or)
  - A validation condition in a Configuration rule can have type ConfigBlock, ConfigText, HostName, NodeModel, SoftwareVersion
- **Condition_logic:** A flag, to check if this is a logic or a validation condition
  - 0: This is a validation condition
  - 1: This is a logic condition
- **Condition_lines:** The lines that the condition has to match for it to be compliant
- **Condition_lines_exclude:** The lines that must not exist in the rule's lines for the condition to be compliant
- **Condition_include:** A flag, to check in what way the condition's lines should be evaluated:
  - 0: The block must contain lines that contain Condition_lines, and those lines can contain more than just that.
  - 1: The block must not contain Condition_lines
  - 2: The block must contain exact matches of the lines of Condition_lines
  - 3: The block must in exact matches of the lines of Condition_lines, and no other lines
- **Condition_order:** A flag, to check whether the condition's lines should match in exact order
  - 0: The condition lines can match in any order
  - 1: The condition lines should match in exact order
- **Condition_regex:** A flag, to check whether the condition's lines can contain regex
  - 0: The condition lines is just plain text
  - 1: The condition lines can contain regex

- **Leading_spaces:** A flag, to check whether the condition's lines will be parsed to take leading spaces in account. Default 0.
    - 0: The condition lines will ignore any leading spaces when parsing
    - 1: The condition lines will parse the exact amount of leading spaces. Config lines that don't have the exact amount of leading spaces are non-compliant.
- **Random_word_order:** A flag, to check whether the condition's lines will be matched with random word order. Default 0.
    - 0: The condition lines will only match if the words are in exact order
    - 1: The condition lines will match any line as long as they contain all of the line's words; order doesn't matter.

Compliance nodes

Policy checks are stored in as a compliance node. Each combination of node to policy is stored separately. The nccmd daemon uses it to check when it should check for compliance and its results are bundled to create reports. A compliance node has the following attributes:

- **Policy_id:** The a link to the policy
- **Policy_schedule_id:** If this compliance node has been scheduled: a link to its policy schedule. This won't be relevant in the initial compliance release and be available for its second phase.
- **Hostname:** The hostname
- **Vendor_type** The node's vendor type
- **Status:** The result of the compliance node's compliance check:
    - 0: Non-compliant
    - 1: Compliant
- **Severity:** The compliance node's severity. Defaults to 0 (Low), 1 (Medium), 2 (Serious), 3 (High), can be edited in the NCCM Lookup under the Rule_severity. Its value is the highest severity of its non-compliant rules. If the compliance node is compliant, this value is a -1.
- **Node_scope:** In which environment does this node exist?
    - 0: This is an yce node
    - 1: This is a cmdb node
- **Schedule_time:** If this value is set, this is the date where this compliance node will be polled for compliance. If it is not set, it won't be picked up by any daemon.
- **Report_id:** Links to the compliance node's report about its latest compliance check.
- **Report_summary:** A one-line summary of the policy check's report.
- **Last_change_date:** The last time this compliance node's status changed from compliant to non-compliant, or reverse.
- **Last_check_date:** The last date this compliance node was checked for compliance.
- **Server:** The server currently checking the policy on this node for compliance. This makes sure that if a server is currently busy with this record, another server cannot also claim it. If it's empty then this compliance node is not currently active and can be claimed by the first server who wants to claim it.
- **Schedule_servers:** A list of all servers that are allowed to check the policy on this node for compliance
- **Policy_group_ids:** A quick link to the node group that couples this policy to this node. In case a node is in two node groups, only one is picked. Which one doesn't matter.
- **Nccm_id:** The latest nccm entry for this node

A few notes: This table does not maintain history. If a compliance check for a policy is performed on a node, it will overwrite the currently existing values. If a node group is removed from a policy, all its

nodes also removed from this table, unless they happen to be present in a different node group. The nccmd daemon will do this, so the deletion won't be instantaneously, but every five minutes the daemon will go through the Compliance node table and clean up the obsolete records. The flags for this are stored in the Nccm Lookup, under the Cmpl_policy_update tweak. Its Str_value will contain a comma-separated list of all policies that need to be evaluated.

In the same way, if you update a policy, its nodes area also automatically re-evaluated for compliance to see if they are still compliant to the policy's new definitions. This also goes for the policy's rules and conditions.

Policy Schedules

Policy schedules are used to schedule a compliance check for a certain time, controlled by the user. Policy schedules will be available in Compliance Phase 2. A policy schedule has the following attributes:

- **Policy_schedule_id:** The policy schedule's id
- **Policy_id:** A link to the policy's id. One policy schedule can have a link to just one policy.
- **Status:** The schedule's status
  - 0: Disabled (it will not be used when scheduling new policies)
  - 1: Enabled
- **Time_interval:** The time interval from which the next schedule time is calculated. There are four types:
  - **Hour of day**: This schedule will be for every day, at certain hours of the day. You can specify multiple hours (separated by a comma, for example 16,20), and even ranges of hours (separated by a dash, for example 4-6). The first and last checkbox are shortcuts for 0:00 and 23:00 respectively.
  - **Every x days**: This schedule will take place every x days. Only one number is allowed, and you can set the schedule time for up to the whole hour.
  - **Day of week**: This will schedule based on the day of the week, denoted by its two-letter abbreviation: mo, tu, we, th, fr, sa and su. Commas to denote multiple days and dashes to denote ranges are also supported. First and last are shortcuts for monday and sunday respectively. You can also set the schedule time for up to the whole hour.
  - **Day of month**: This will schedule based on the day of the month, from 1 to 31, where non-existing days like February 30th will be ignored. Commas to denote multiple days and dashes to denote ranges are also supported. First is a shortcut for the first day of the month, and last for the last, depending on what month it is. You can also set the schedule time for up to the whole hour.
- **Repeating:** Whether a schedule is repeating (it will keep being scheduled) or not (it will only be scheduled once)
  - 1: Repeating
  - 0: Non-repeating (It will be disabled after being scheduled)

Command replies

Command replies store the result of a command, defined as a command rule. There is no history in this table. Here are its attributes:

- **Cmd_reply_id:** The reply's id

- **Hostname:** The hostname on which the reply has been executed
- **Policy_id:** The command rule's policy id
- **Rule_id:** The id of the command rule that the command originated from
- **Response_text:** Contains in plain text the result of the command on the node.
- **Variables:** Currently not used
- **Parse_error:** Shows if there was an error while running the command:
  - 1: There was an error. Rules that use this reply will automatically be non-compliant.
  - 0: Everything is okay.