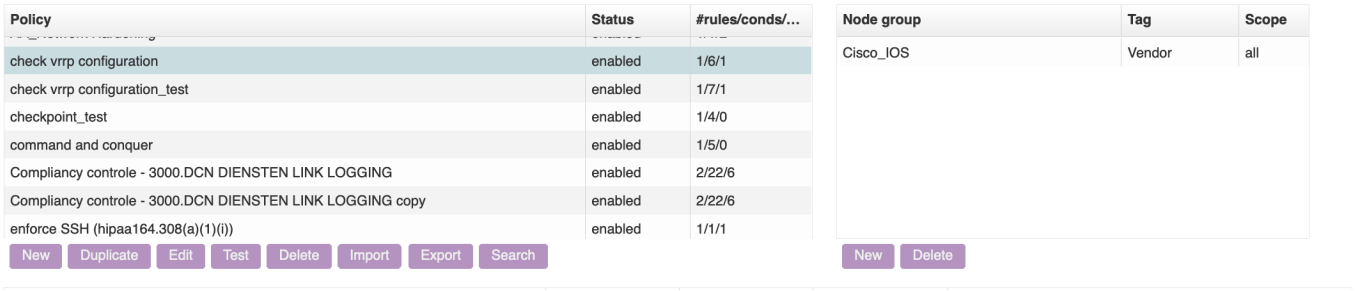
Table of Contents

Policies

In this form you can create, test and manage compliance policies, rules and conditions. A policy is tested on a node, to return either compliant or non compliant. A policy contains various vendor type specific rules, each of these rules contains a string of conditions that together form a logic the rule tests at. Only if all rules for the node's vendor type are compliant, is the policy compliant for this node.

Policies

The policy grid:



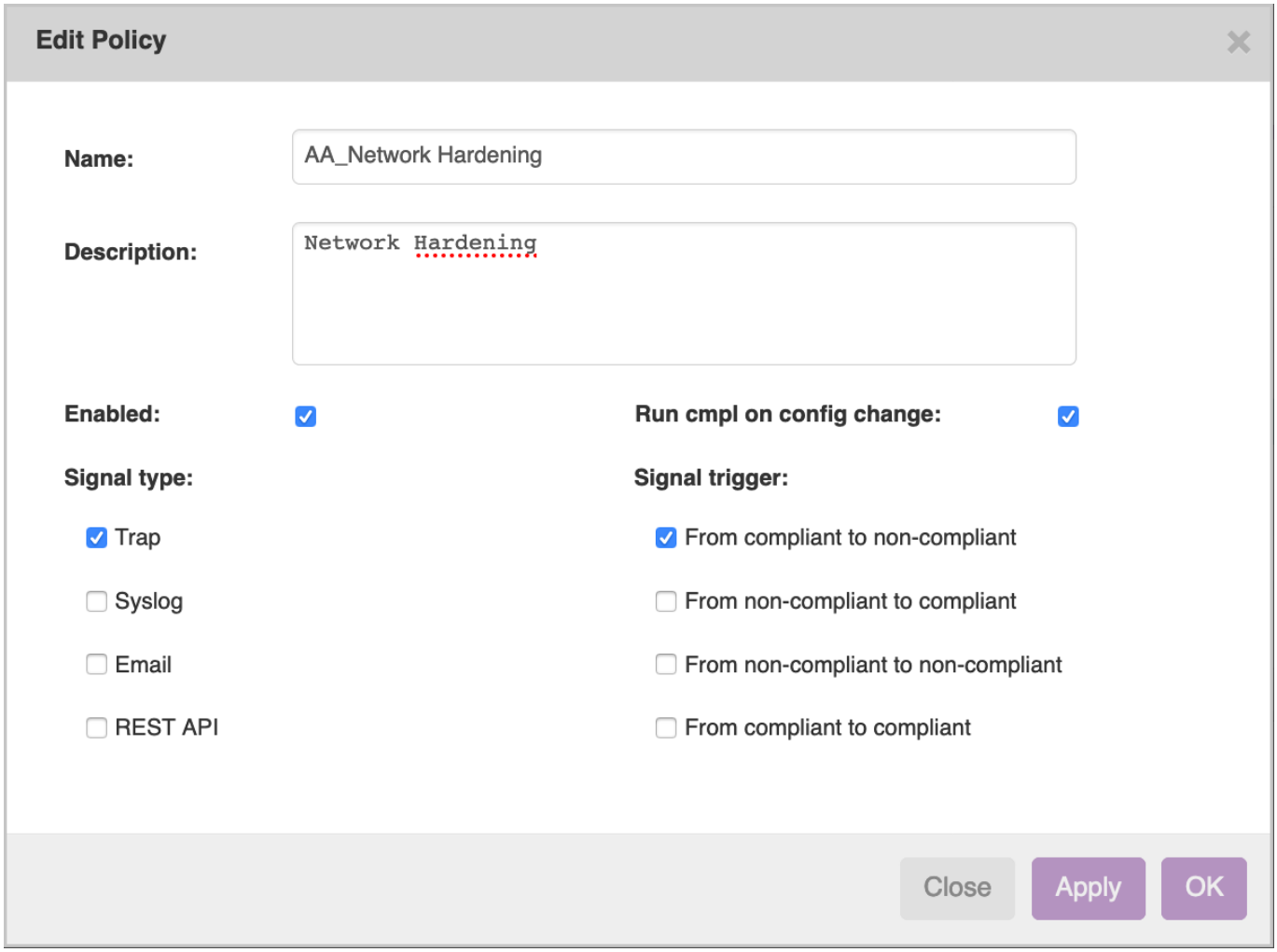
You can create a new policy, edit an existing one and you can duplicate it. The copy will contain all of the policy's rules, conditions and node groups. You can test a policy on a node to see whether the logic works and search through all policies you have. You can delete, export and import them. You can export single or multiple policies. The import function works both with exported files from netYCE and HPNA.

A few caveats about importing from HPNA:

- a rule with a vendor type not supported by netYCE will not work, a list of supported vendors can be found here
- diagnostics rules are not supported. NetYCE's version of this are command rules, available with Phase 2 of compliance, but they will not support a conversion from diagnostic rules.

The grid on the right contains all node groups linked to this policy. All nodes belonging to these node groups will be validated on compliance, unless the policy in question is disabled (you can see this on the status-column of the policy grid). The scope can be either 'cmdb', 'yce' or 'all', meaning where the node groups should be evaluated from. For more details on node groups, see Node Groups.

Edit Policy



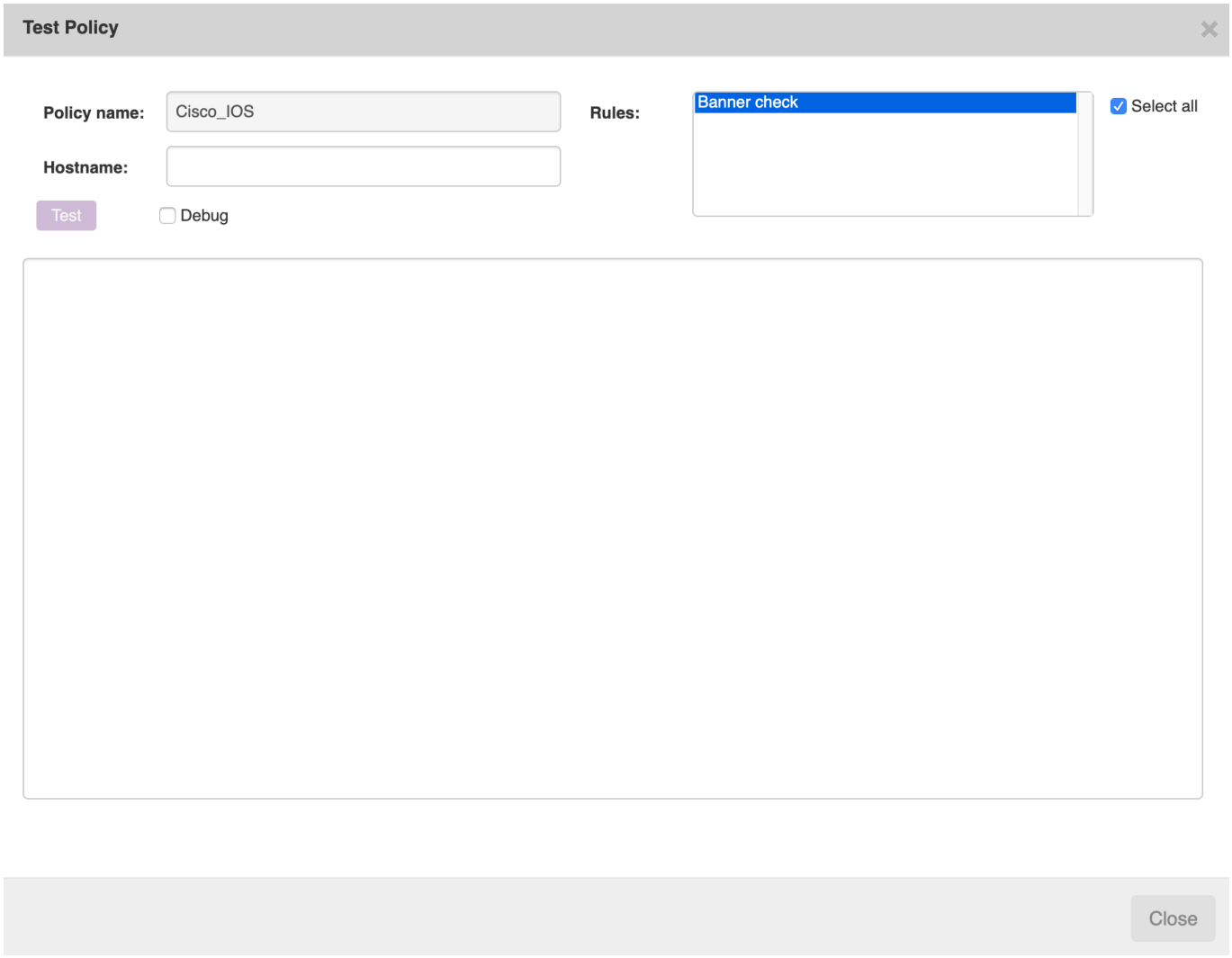
The options (Trap, Syslog, Email and REST API) presented under Signal type represent the available actions to be undertaken upon the compliance result. This can be triggered in four different ways:

- From compliant to non-compliant
- From non-compliant to compliant
- From non-compliant to non-compliant
- From compliant to compliant

The most important option is whenever a node changes from compliant to non-compliant.

You can enable or disable a policy here with its checkbox *Enabled*. “Run cmpl on config change” means that whenever we detect a config change, the node will be scheduled for a compliance check. This is useful for nodes whose configs change once in a while. For nodes whose config change constantly it is better to use scheduled policies. This is a work in progress and will be released in future versions of netYCE.

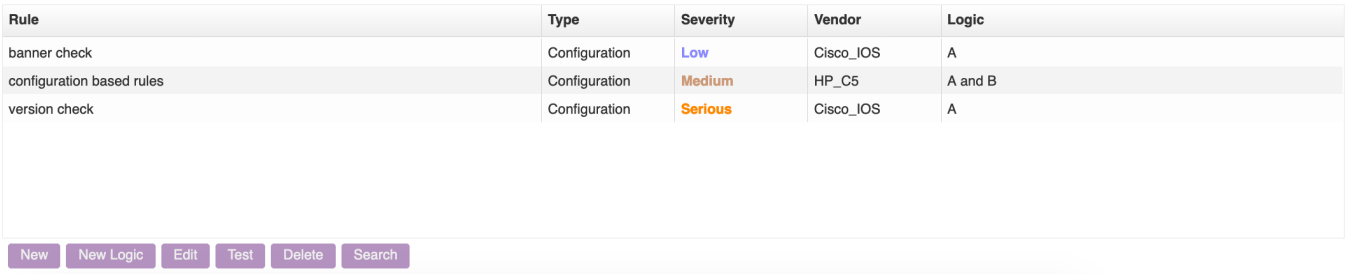
Test



Policies can get quite complex, and testing them is difficult when you're working with live nodes and a daemon that only runs periodically. This form is meant to give you a quick option to test whether your policies do what they're supposed to do. You can select multiple rules (by not selecting any it just takes all of them) belonging to this policy, enter a hostname and they will be evaluated. The debug-check shows more detailed information that might help you in case there is a problem.

An important caveat is this: the form checks the policy on the config of the node that's saved in the NCCM, it will not poll the node whatsoever. This means that for this to work, you do need to have an NCCM config for the node in the database, otherwise this function will not work.

Rules



Rule	Type	Severity	Vendor	Logic
banner check	Configuration	Low	Cisco_IOS	A
configuration based rules	Configuration	Medium	HP_C5	A and B
version check	Configuration	Serious	Cisco_IOS	A

New | New Logic | Edit | Test | Delete | Search

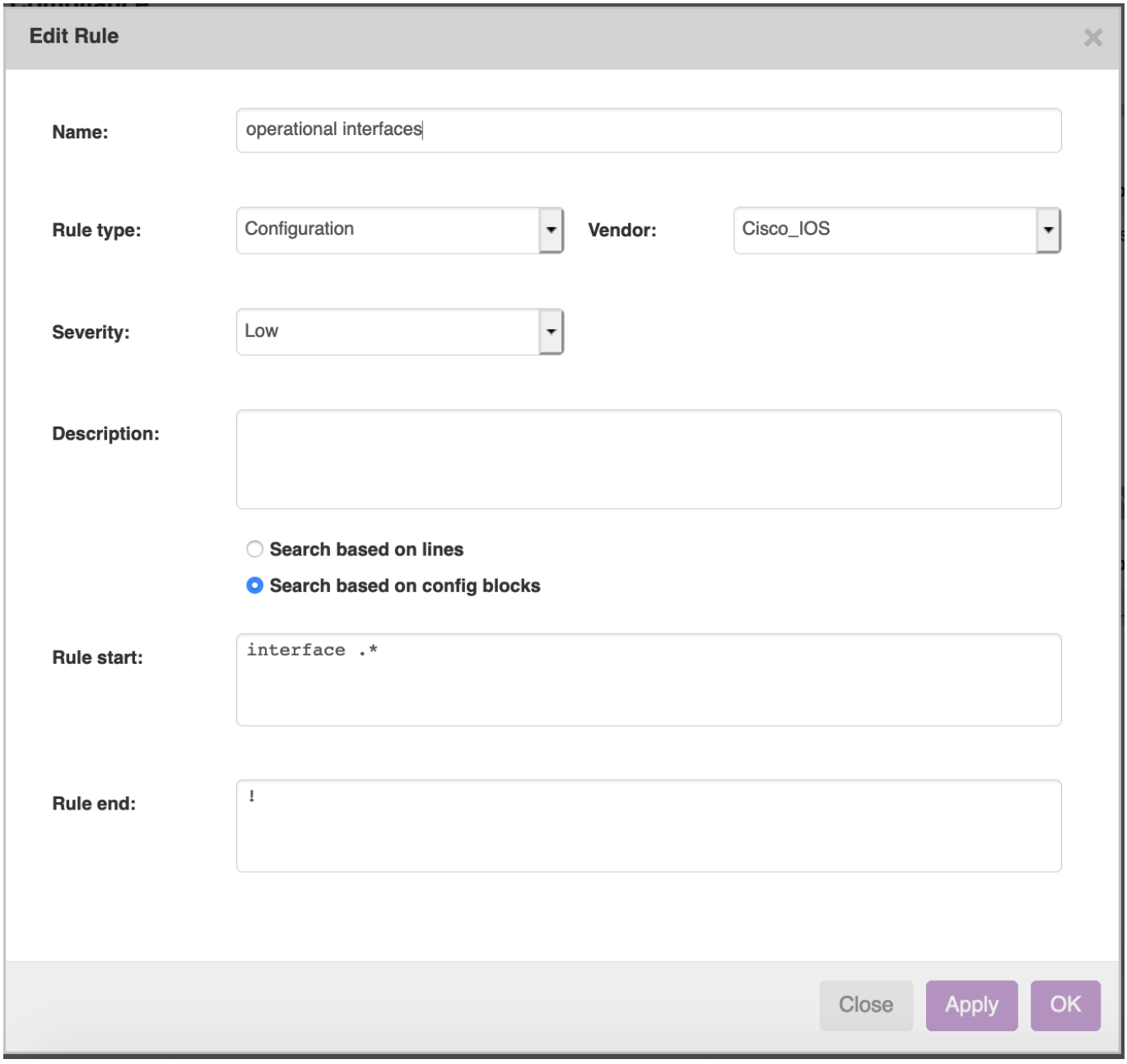
Rules are vendor type bound: only for nodes with that vendor type will they be evaluated. There are

three types of rules: Configuration, Command and Multi-config.

- Configuration rules take a config or a part of the config (as signified by Rule_start and Rule_end) and runs a number of conditions on it.
- Command rules take the output of a command on the node, and compares it to a number of conditions. Future plans are also to parse these into variables using command parsing. This feature is not yet available and will be in future netYCE releases.
- Multi-config rules compare the config to the config of one to three other nodes, to see if they are equal. For more information, refer to the Compliance user guide

You can create, edit and delete nodes and search through them. A rule's conditions combine together with a 'logic'. You can test whether this logic has a valid syntax and create a bunch of conditions at the same time with the 'new logic' button, but more on that below, in the 'Conditions' section.

Edit Rule



Rules can parse config based on lines, or on blocks. To explain config blocks: a config consists out of a number of text blocks. Think of a block as follows:

```
block head
 block body
 block body
 block body
!
```

Or:

```
block a
block b
block c
block d
```

Or even hierarchical:

```
block head
 block body
 block body
 block body
 subblock head
  subblock body
  subblock body
  subblock body
 block body
!
```

Rule_start looks at these block heads, and returns all blocks where they match. If you want to be extra specific, you can use Rule_end to further filter blocks based on their last line. If no Rule_start is specified, the whole config is taken.

Some vendors, Juniper for instance, have complex hierarchical trees within their blocks. For hierarchical blocks we also support the full paths that they can be found at. For example when we look at the above hierarchical block, a Rule_start of:

```
block head subblock head
```

Will work. The other case where all lines are at the same indentation and there isn't really a 'head', the path becomes simply the words they have all in common. So for the above example the Rule_start of:

```
block
```

will work.

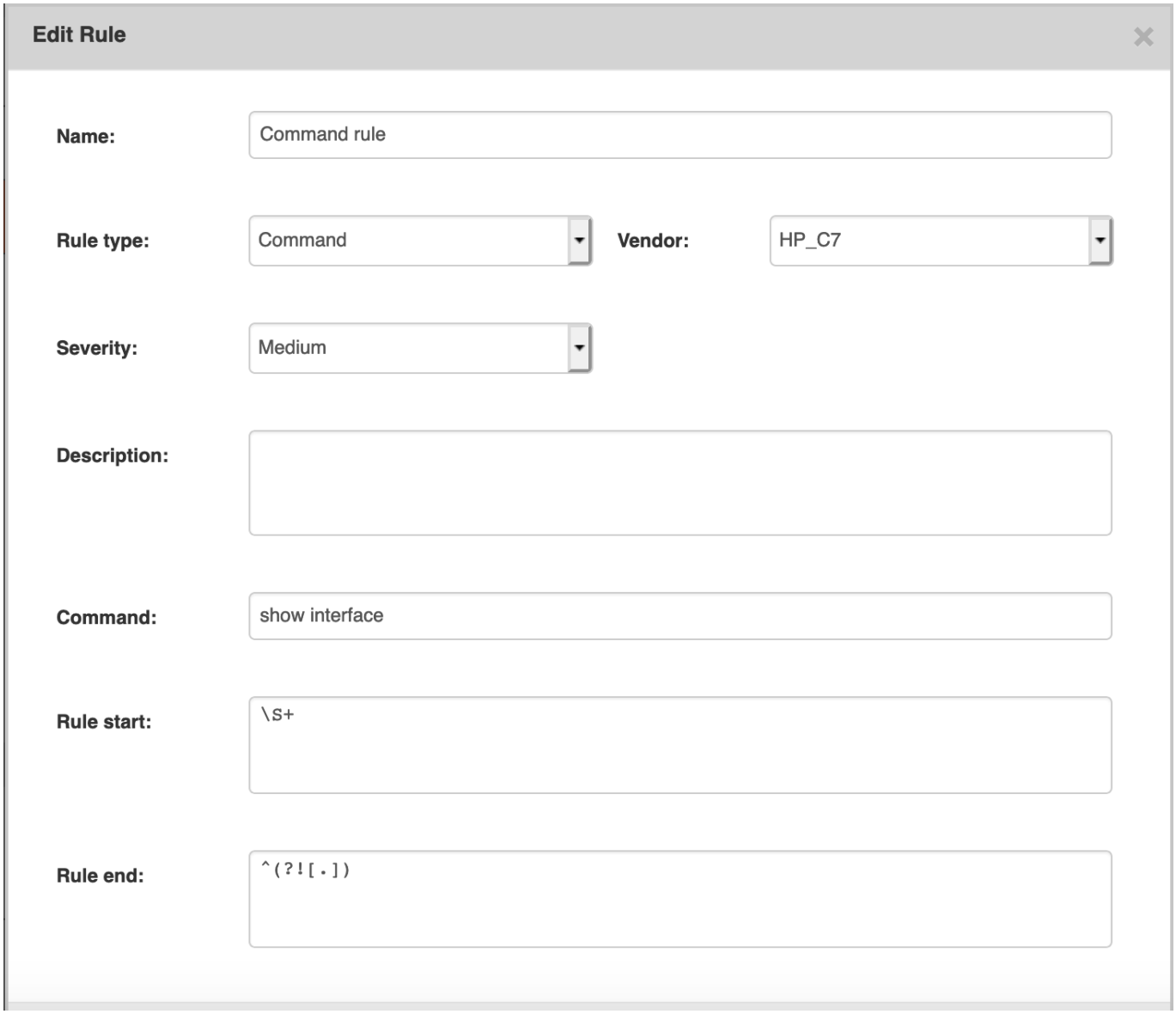
Sometimes however, you will want to run compliance on a string of text that does not fit well inside these blocks. In that case you can check the option to parse by lines. In that case the config is parsed, and all lines inbetween and including the rule start and the rule end will match. Multiple blocks that match these criterias will all match. If no rule end is defined, it takes the whole config starting from

the first line that matches the rule start. If no rule start or end is defined, it still takes the whole config.

Command Rules

This is a feature that will be available with the release of Compliance Phase 2.

Command rules allow you to check the result of a command for compliance. The form is nearly identical for normal rules, with the addition of a Command-field:



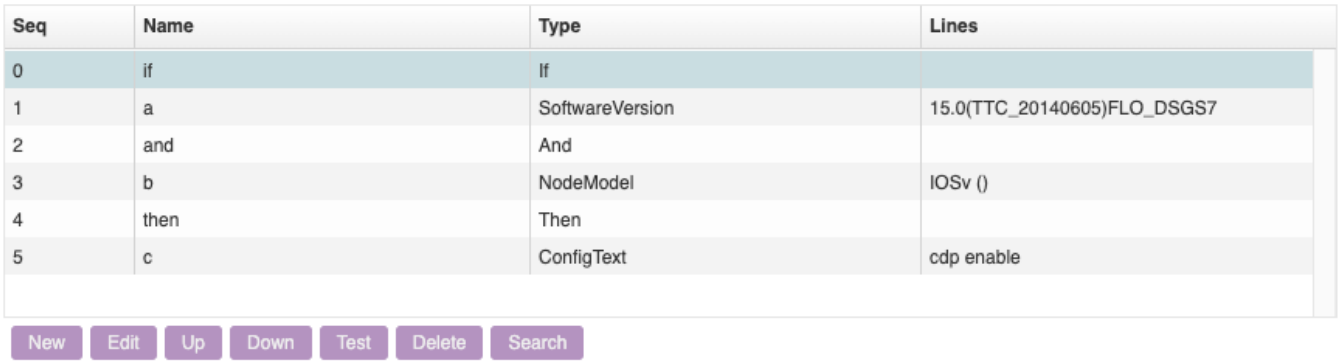
This is the command that will be run on the server.

Like all rules, these are limited per vendor type, so in this example, this rule will only be checked for HP C7 nodes.

Rule start and Rule end can be used to parse part of the reply. Since commands are easier to parse than entire config, these values can be plain text, or contain regular expressions. When using both a rule start and rule end, multiple blocks of text can be matched. Compliance will then be run on all blocks separately, like with configuration rules.

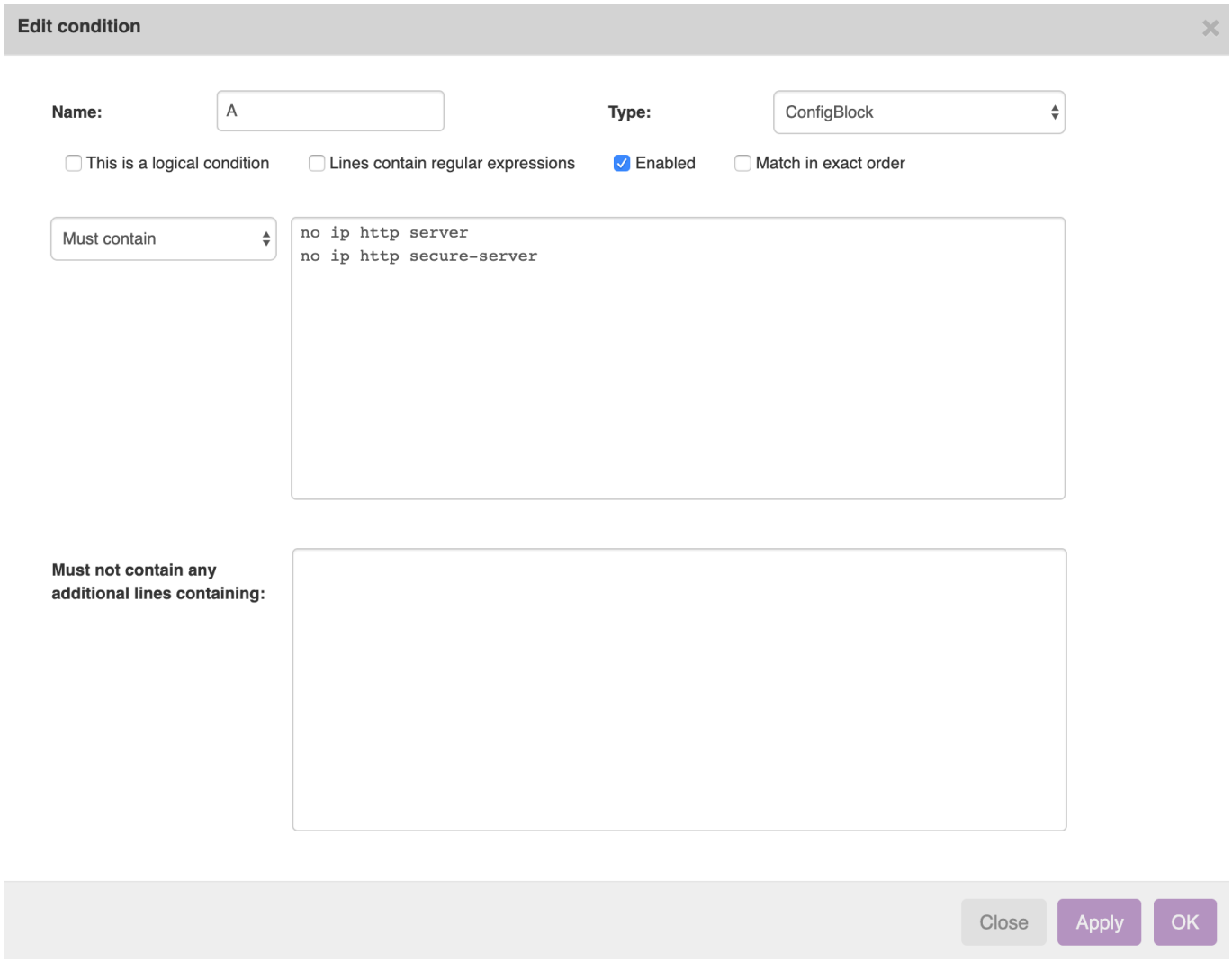
When you finish editing a command rule, the nodes that are linked to it are automatically scheduled for an nccm poll, following a compliance check. It will take the nccmd daemon a few minutes to get to it, but its update process is automated.

Conditions



There are two types of conditions: Conditions that test a config block, and logic conditions that tie everything together. Logic conditions can be either 'if' ('then', 'else'), 'and', 'or' and '(' and ')' to group things together. Together they form a logic string that needs to be valid for a rule to be compliant. A rule won't work if the syntax of its conditions is invalid, you can use the 'test'-button at the rules-grid to verify this.

Edit condition



You can switch between condition types using the “This is a logical condition”-checkbox. Like policies, conditions can also be enabled or disabled. A disabled condition will not be evaluated from a rule's logic and compliance checks. Disabled conditions will also be greyed out in the conditions grid in the main compliance form.

You can select different condition types, and this determines what text will be used for the condition:

- **ConfigBlock:** The config block as dictated by the Rule_start and Rule_end of the condition's rule
- **ConfigText:** The whole config
- **NodeModel:** The node's node model, as retrieved from cli command output
- **SoftwareVersion:** The node's software version, as retrieved from cli command output
- **Hostname:** The node's hostname

This string of text will be compared to the condition's lines. There are four ways to compare:

- **Must contain:** For each condition line, the text needs to contain a line that matches it. Lines don't have to match exactly, as long as the condition's line is part of it.
- **Must contain except:** There should be no instance of any condition line in the text.
- **Must contain only:** For each condition line, there needs to be a line in the text that matches it exactly
- **Must contain exclusively:** The text should match each condition line, and there should be no other lines present

Additionally you can specify if the lines should contain regex or not. If the field “Must not contain any additional lines containing” is filled, the text will also be checked to see if there aren't any other additional matches, beyond the lines that have already matched.

Starting from compliance phase 2, you can also parse relations and variables. For example, a line:

```
hostname <node>
```

Will parse <node> as the node's hostname. Relations will also be parsed. For more information, refer to the Relations reference.