



Configuration Compliance

version 10.6

Table of Contents

Table of Contents	1
Configuration Compliance	3
Policies	5
Use Cases	8
Scenario 1: IP Domain Name	8
Configure Policy	9
Configure Rules	9
Basic Information	10
Platform Selection	10
Rule Variables	11
Conditions and Actions	11
Condition Details	12
Test Config	13
Action Details	14
Scenario 2: NTP Server configuration check	17
Condition1	18
Condition2	19
Scenario 3: Interface configuration check	22
Scenario 4: Enforce VTY Session Timeouts	26
Scenario 5: Enforce OSPF Router Id as Loopback0	30
Condition1	32
Condition2	36
Scenario 6: BGP TTL Hop-count	38
Condition1	40
Condition2	43
YANG Compliance	44
Policy creation with Xpath Expressions	44
Few examples	45
Scenario 7: IP Domain Name	45
Scenario 8: IP Name-server check	47
Scenario 9 : NTP server Check	48
Scenario 10 : Interface Check with rule_variable	50
Scenario 11 : VRF Check with rule_variable	53
How to derive the X-path expressions	56

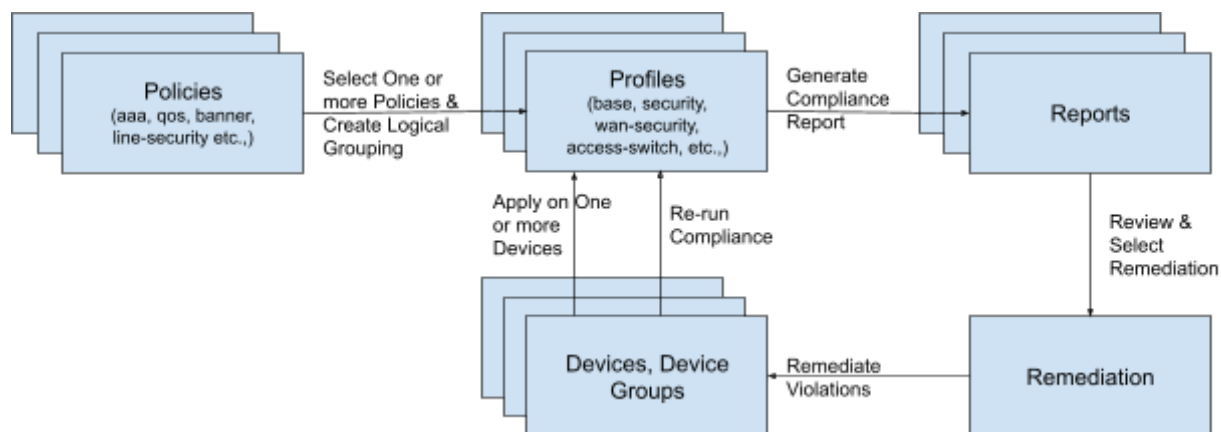
Policy creation with XML Template Payload	57
Few examples	57
Scenario 12 : IP Domain name check	57
Scenario 13 : IP Name Server check	60
Scenario 14 : Interface check	62
Scenario 15 : VRF check	65
How to derive the XML Template payload	69
Profiles	70
Report	73
Pivot by device	77
Pivot by policy	78
Pivot by device type	79
Pivot by location	80
Pivot by device group	82
Pivot view	83
CRV(conditional Report View)	84
Remediation	85
Bulk delete	88
Export	89
Archive	89
Dashboard	90
FAQs	94
Appendix	95

Configuration Compliance

Configuration Compliance feature allows users to Define & Enforce Configuration Compliance Standards. This is realized within ATOM using the following primitives.

1. Policies - Define Configuration standards & Remediation Policy by Device Family, Device Type, OS Type etc.,
2. Profiles - Group multiple policies and apply configuration standards on one or more devices
3. Reports - Comprehensive compliance reporting view at device level
4. Remediation - Fix Policy Violations on one or more devices

Following diagram summarizes the overall flow.



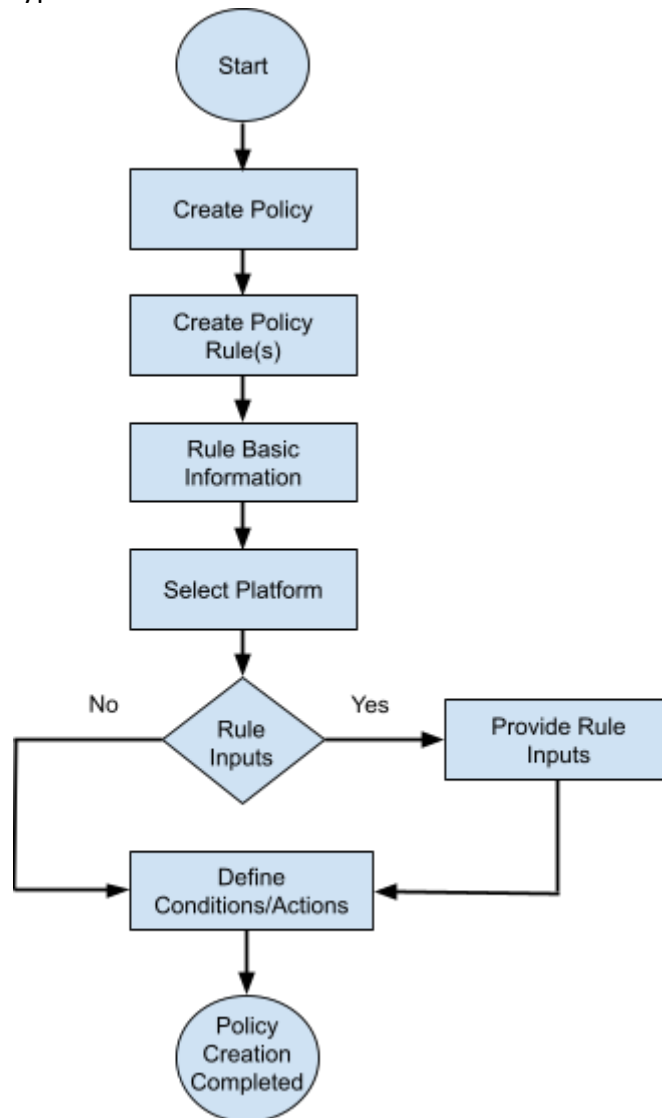
ATOM supports Configuration Compliance for the following Vendors:

- Cisco Systems
- Juniper Networks
- Fortinet
- Force10 Networks
- Brocade
- PaloAlto Networks
- Riverbed Technology
- F5 Networks

Policies

Compliance policy allows configuration standards to be defined in CLI format and YANG format(x-path or xml). Following provides a high level overview of a Policy:

- Policy is a collection of Rules
- A Rule contains one or more Conditions
- Condition describes
 - Expected Configuration. Configuration can be parameterized through Rule Variables.
 - Action to be taken on a condition evaluation includes CLI commands or Netconf XML RPC format to be used to remediate a violation.
- A Rule can be attached to one or More device platforms - Vendor, OS Type, Device Family, Device Type and OS Version



Use Cases

#	Configuration Standard Style	Example	Reference
1	Static Configuration	<p>Example: All Devices in Target Network should Contain a specific Domain Name</p> <p>Expected Configuration:</p> <pre>ip domain-name anutacorp.com</pre> <p>Fix Configuration:</p> <pre><<If missing, configure the above command>></pre>	Scenario1
	X-path Expression	<p>Xpath Expression:</p> <p>Cisco-IOS-XR-native:native/ip/domain/name=`anutacorp.com`</p>	Scenario6
	XML Template Payload	<p>Template Payload:</p> <pre><native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native" > <ip> <domain> <name>net.disney.com</name> </domain> </ip> </native></pre>	Scenario11
2	Dynamic Configuration with User provided values	<p>Example: Devices in Target Network should have a specific Loopback interface - Loopback0 or Loopback1 based on user input.</p> <p>Expected Configuration:</p> <pre>interface {{ interface_name }}</pre> <p>Fix Configuration:</p> <pre><<If missing, Configure the specific Loopback interfaces>></pre>	Scenario3

	X-path Expression	Xpath Expression: Cisco-IOS-XE-native:native/interface/Loopback/name='0' and Cisco-IOS-XE-native:native/interface/Loopback[name=0]/ip/address/primary/address='{{ lo0_ipv4addr }}' and Cisco-IOS-XE-native:native/interface/Loopback[name=0]/ip/address/primary/mask='255.255.255.255' and Cisco-IOS-XE-native:native/interface/Loopback[name=0]/ipv6/address/prefix-list/prefix='{{ lo0_ipv6addr }}'	Scenario9
	XML Template Payload	Template Payload: <pre> <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native" > <interface> <Loopback> <ip> <address> <primary> <address>10.100.99.98</address> <mask>255.255.255.255</mask> </primary> </address> </ip> <ipv6> <address> <prefix-list> <prefix>2605:30C0::3B/128</prefix> </prefix-list> </address> </ipv6> <name>0</name> </Loopback> </interface> </native> </pre>	Scenario13
3	Configuration with Patterns, Wildcards,	Example: All the VTY lines should have specific exec-timeout and session-timeout configured.	Scenario4

	etc.that require Regular expressions	Expected Configuration: <pre>line vty (.*) session-timeout 10 exec-timeout 10 0</pre> Fix Configuration: <<If missing, Configure the timeouts under all matching VTY lines>>	
4	Configuration with sub-modes	<p>Example: The physical Interface should not be shut down and show be in auto-negotiation mode</p> Expected Configuration: <pre>interface {{ interface_name }} no shutdown negotiation auto</pre> Fix Configuration: <<If missing, Configure the above commands for one or more interfaces>>	Scenario3
5	Removing unwanted extra configuration	<p>Example: Finding the Devices having extra ntp-server addresses configured and removing those other than expected server addresses.</p> Expected Configuration: <pre>ntp-server 10.1.1.1</pre> Fix Configuration: << Configure above ntp-server if not found. Remove any ntp server other than 10.1.1.1 >>	Scenario2
6	Advanced: Presence of an entity value from one block in another	<p>Example: Finding the devices in the network which doesn't contain the OSPF router-id configured as per loopback0 ip address.</p> Expected Configuration: <pre>interface Loopback0 ip address 45.45.45.5 255.255.255.255 ! router ospf 100</pre>	Scenario5

		<pre>router-id 45.45.45.5</pre> <p>Fix Configuration:</p> <pre>router ospf 100 router-id 45.45.45.5</pre>	
--	--	--	--

Scenario 1: IP Domain Name

Scenario: Network Devices must have domain name configured. In this example we are looking for the domain name as **anutacorp.com** across all devices in the lab.

Platform:

Cisco IOS-XE

Expected Configuration:

```
ip domain-name anutacorp.com
```

Fix-CLI Configuration:

```
ip domain-name anutacorp.com
```

Follow the steps below to configure Compliance Policy for the above scenario.

1. Configure Policy

Steps:

- Navigate to Resource Manager > Config Compliance -> Policies
- Click '+' to create new Policy and provide the following information
 - Policy Name
 - Description

Add Policy

●-mandatory information

Policy Name ●

Policy name. Can contain AlphaNumerics and underscore characters o...

IP_Domain_Name

Description

Description of the policy

Check whether ip domain name is present in the device or not

Create Policy

2. Configure Rules

One or more rules can be configured to express configuration standards. Based on the complexity of the scenario, configuration standards can be broken up into more than one condition.

Steps:

- Navigate to Resource Manager > Config Compliance -> Policies
- Create/Select a Policy
- Click '+' to create new Rule
- ATOM opens up new wizard as shown below

Add Rule

Basic Information Platform Selection Rule Variables Conditions and Actions

- Rule has four components
 - Basic Information
 - Platform Selection
 - Rule Variables
 - Conditions & Actions

Basic Information

Provide basic information as described below. Information provided here is for documentation purposes only.

Rule Name: Provide any Name

Description: Brief explanation of the configuration evaluation that the rule is going to perform.

Impact: If the device configuration does not meet the rule or rules in the policy, type it in the Impact field.

Suggested Fix: Using which non-compliance can be corrected and device returns to a state of Compliance.

Edit Policy | [IP_Domain_Name](#)

Add Rule

Basic Information

Platform Selection

Rule Name

Check_Ip_Domain_Name

Description

Check domain name for Cisco devices

Impact

If domain name is not present in the device. Device will be non-compliant.

Suggested fix

ip domain name anutacorp.com

Create Rule

Platform Selection

Rules contain configuration standards expressed in CLI Configuration format. Configuration standard can be at Vendor level, Device Type, Device Platform, OS Type or OS Version.

Steps:

- Navigate to Config Manager > Config Compliance -> Policies -> Rules
- Create/Select a Rule & Provide the following information
 - Vendor
 - OS type
 - Device Family
 - Device Type
 - OS Version

Add Platform

Vendors ●
Cisco Systems

OS type
Select

Device family
Select

Device type
Select

OS version
ALL

Note: Platform Selection will be used during Policy execution. Devices that don't match the above criteria are skipped.

Note: It's not common to have more than one Platform

Rule Variables

Rule variables allow configuration to be parameterized.

Steps:

- Navigate to Resource Manager > Config Compliance -> Policies -> Rules
- Create/Select a Rule Variables
 - Key - Provide unique name to identify rule variable
 - Description - Describe rule-input configuration
 - Default Value - Default value. Can be overridden during Policy execution time

Conditions and Actions

Expected configuration & actions to be taken when violations are detected are specified in the *Conditions & Actions* section.

Based on the complexity of the scenario, configuration standards can be broken up into more than one condition.

Steps:

- Navigate to Resource Manager > Config Compliance -> Policies -> Rules
- Create/Select Conditions & Actions
 - Condition Details - Described Below
 - Action Details - Described Below

Condition Details

Condition section provides users to specify the expected configuration and various options on how to match the expected configuration including option to identify sub mode configuration blocks.

Condition Name: Name of the Condition

Sequence Number : Order of the condition execution.

Scope Details

Condition scope details: Scope could be either full configuration copy or configuration matched in prior condition.

- *Configuration* - Full Configuration
- *Previously_Matched_Blocks* - Subset of configuration matched by prior condition

Block Options

Start Expression - Regular expression indicating the start of the sub-block.

End Expression - Regular expression indicating the end of the sub-block.

Condition Match Criteria

Operator:

MATCHES_THE_EXPRESSION - Checks whether the condition value exactly matches with device configuration or not.

DOESNOT_MATCHES_THE_EXPRESSION - Checks whether the condition value does not match with the device configuration or not.

CONTAINS_STRING - Checks whether the device configuration contains condition value config or not.

Rule-Pass-Criteria:

All_SubBlocks - Checks whether the condition value matches in all the blocks or not.

Any_SubBlock - Checks whether the condition value matches in any of the blocks or not.

Value: Value field accepts Configuration Standard as CLI Configuration. Following types of configuration can be provided:

- Static Configuration
- Dynamic/Parameterized Configuration
- Configuration with Regular Expressions
- Configuration coupled with Jinja2 Templating

Note: For some Vendor Configurations like Cisco IOS-Style, whitespace in command prefix is mandatory to identify commands at sub-mode level.

For Scenario 1 - Provide value as **ip domain-name anutacorp.com** to search for a given domain name in the running configuration.

Test Config

Based on the complexity of the configuration standard, Value may be complex and may need to build up iteratively. Test Config utility helps the CLI configuration condition to be validated against Test Configuration.

Steps:

- Navigate to Resource Manager > Config Compliance -> Policies -> Rules ->
- Create/Select Conditions & Actions
- Click “Launch Test Config” will launch a form to Test Condition

Condition Match Operator:

MATCHES_THE_EXPRESSION
 DOESNOT_MATCHES_THE_EXPRESSION
 CONTAINS_STRING

Value: Sample configuration to be tested. Value will be shown from the Condition Details. Value can be further refined

Note: Any Edits to Value will reflect in the Condition Details -> Value and Vice-versa.

Test Configuration: Sample device configuration

Rule Variables: The rule variables created in the rule will be shown here with default values. Values can be modified.

Note: Any Edits to Values will not be reflected in the Rule Variable default values provided in Rule Variables Section.

Test Results: Based on Condition Match Operator test results will be shown on the right hand side

Action Details

Action can be taken after Condition evaluation. Condition can result in either a “Match” or “Non-Match”. Depending on the scenario one or both criteria may apply.

Select Match Action - This option is applicable when Condition evaluates to a Match

Select action:

Continue - continue execution to next condition

Donot_raise_violation - skip execution and don't raise violation

Raise_violation_and_continue - raise violation and continue execution to next conditions

Raise_violation - raise violation and skip execution

For Scenario 1, no action needs to be taken during a match condition, so select **continue** as action.

Violation severity:

LOW

MEDIUM

HIGH

CRITICAL

Violation message type:

Default_violation_message

User_defined_violation_message

Derive fix cli commands:

Use_unmatched_block - unmatched config from the block

Use_matched_block - matched config from the block

Use_complete_block - total block config

Select Non-Match Action

Select action:

Continue

Don't raise violation

Raise violation and continue

Raise violation

For Scenario 1, Action is required when Condition is not matched. Select **Raise violation and continue**.

Violation severity:

LOW
MEDIUM
HIGH
CRITICAL

Violation message type:

Default_violation_message
User_defined_violation_message

Fix CLI: Provide the CLI Configuration to be used for remediation. Fix CLI can be either provided here or derived.

Option - 1 - Explicit Remediation / Fix CLI

For Scenario 1, Provide “ip domain-name anutacorp.com” in Fix CLI.

The screenshot shows the 'Conditions and Actions' configuration window. The 'Action Details' tab is selected. On the left, under 'Select Match Action', the 'Select action' dropdown is set to 'continue'. On the right, under 'Select Non-Match Action', the 'Select action' dropdown is set to 'Raise_violation'. Below this, the 'Violation severity' dropdown is set to 'CRITICAL', and the 'Violation message type' dropdown is set to 'Default_violation_message'. The 'Fix CLI' text area contains the text 'ip domain-name anutacorp.com'. At the bottom, the 'Derive fix cli commands' dropdown is set to 'Select'.

Option - 2 - Remediation Commands can be derived from Condition evaluation.

Derive fix cli commands:

Options below:

Use_unmatched_block
Use_matched_block
Use_complete_block

For Scenario 1, Select “Use_unmatched_block”. Since this is non-match Action, unmatched_block will be Condition Details->Value and can be used as Fix CLI.

The screenshot shows the 'Conditions and Actions' configuration window. The 'Action Details' tab is selected. On the left, under 'Select Match Action', the 'continue' option is chosen. On the right, under 'Select Non-Match Action', the 'Raise_violation_and_continue' option is selected. Below this, 'Violation severity' is set to 'HIGH' and 'Violation message type' is set to 'Default_violation_message'. There is a 'Fix CLI' text area which is currently empty. At the bottom, 'Derive fix cli commands' is set to 'use-unmatched-block'.

Scenario 2: NTP Server configuration check

Scenario:

1. All devices in the network should contain the designated ntp server.
2. Remove all other ntp servers
3. In this example
 - a. Expected ntp-server = 10.0.0.1

Platform:

Cisco IOS-XE

Expected Configuration:

```
ntp server 10.0.0.1
```

Fix-CLI Configuration:

```
ntp server 10.0.0.1
<<Remove Any Other ntp server other than 10.0.0.1>>
```

This use case uses regular expressions and contains two conditions.

1. Condition-1 - Check for expected config & if not found remediate using Fix CLI.

Fix-cli Configuration :

```
ntp server 10.0.0.1
```

2. Condition 2- Check for unwanted ntp-servers and remove them.

Fix-cli Configuration :

```
no ntp server 10.0.0.2 //Derived
no ntp server 10.0.0.3 //Derived
```

Steps:

- Navigate to Resource Manager > Config Compliance -> Policies
- Click '+' to create new Policy and provide the following information
 - Policy Name - NTP_Common_Peer_Configuration
 - Description
- Select the Policy and Click '+' to create new Rule
 - Rule Name - Check_NTP_Common_Peer_Configuration
- Navigate to Config Manager > Config Compliance -> Policies -> Rules
- Select a Rule & Provide the following information
 - Vendor - Cisco Systems
 - OS type - IOSXE
 - Device Family - ALL
 - Device Type - ALL
 - OS Version - ALL
- Rule variables are not required for this scenario.
- Now fill the Conditions and Actions

Condition1

The Verify_NTP condition will check if the NTP server config is present in the device or not.

Conditions and Actions

Condition Details
Action Details

✕
✓

Condition Name ●

Condition Name, can contain Alphanumerics, underscore, space and hyphen ...

Verify_NTP

Sequence Number ●

Sequence Number controls the order of execution of the Conditions.

1

Scope Details

Condition scope details

Configuration

✱

Block Options

Start Expression

✱

Condition Match Criteria

Operator

MATCHES_THE_EXPRESSION

Rule-pass-criteria

All_SubBlocks

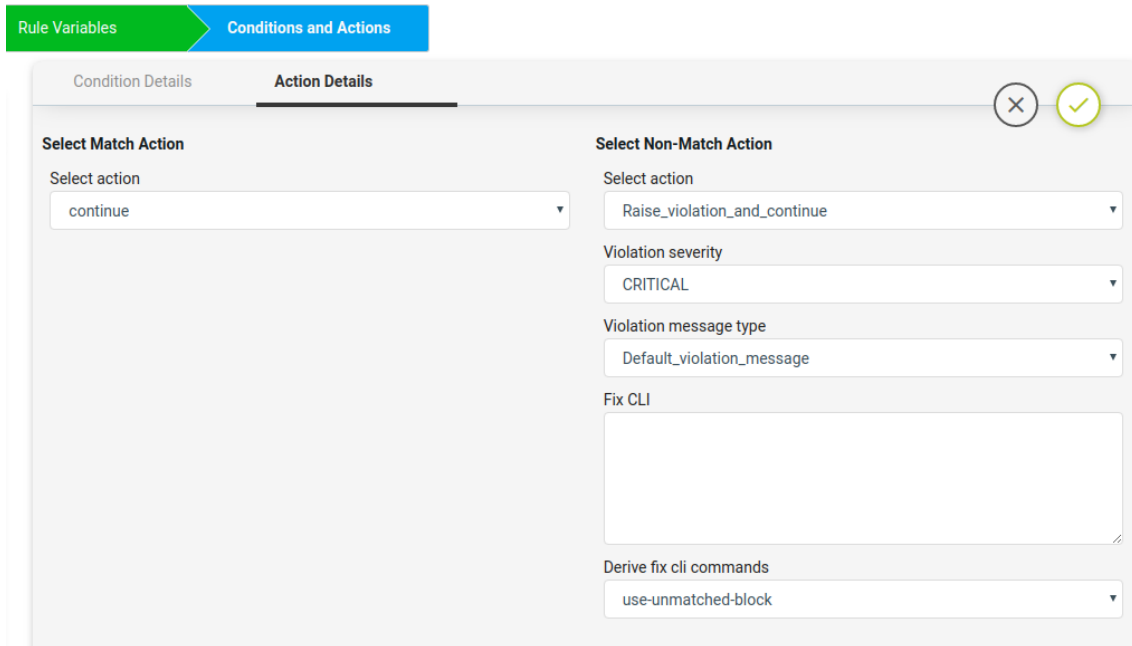
Value ●

ntp server 10.0.0.1

Launch Test Config

Here Non-Match Action can be done either using the commands in Fix CLI or using the Derive fix cli commands.

- Using the Fix CLI user needs to provide the configuration commands manually.
- Using the Derive Fix CLI Commands user needs to select the use_unmatched_block as shown below.



The screenshot shows the 'Action Details' tab in the ATOM Configuration Compliance Guide. It features two main sections: 'Select Match Action' and 'Select Non-Match Action'. In the 'Select Match Action' section, the 'Select action' dropdown is set to 'continue'. In the 'Select Non-Match Action' section, the 'Select action' dropdown is set to 'Raise_violation_and_continue', the 'Violation severity' dropdown is set to 'CRITICAL', the 'Violation message type' dropdown is set to 'Default_violation_message', the 'Fix CLI' text area is empty, and the 'Derive fix cli commands' dropdown is set to 'use-unmatched-block'.

Here on Match Action it will Continue and on Non-Match Action the Derive fix cli commands uses the use-unmatched-block to remediate the device.

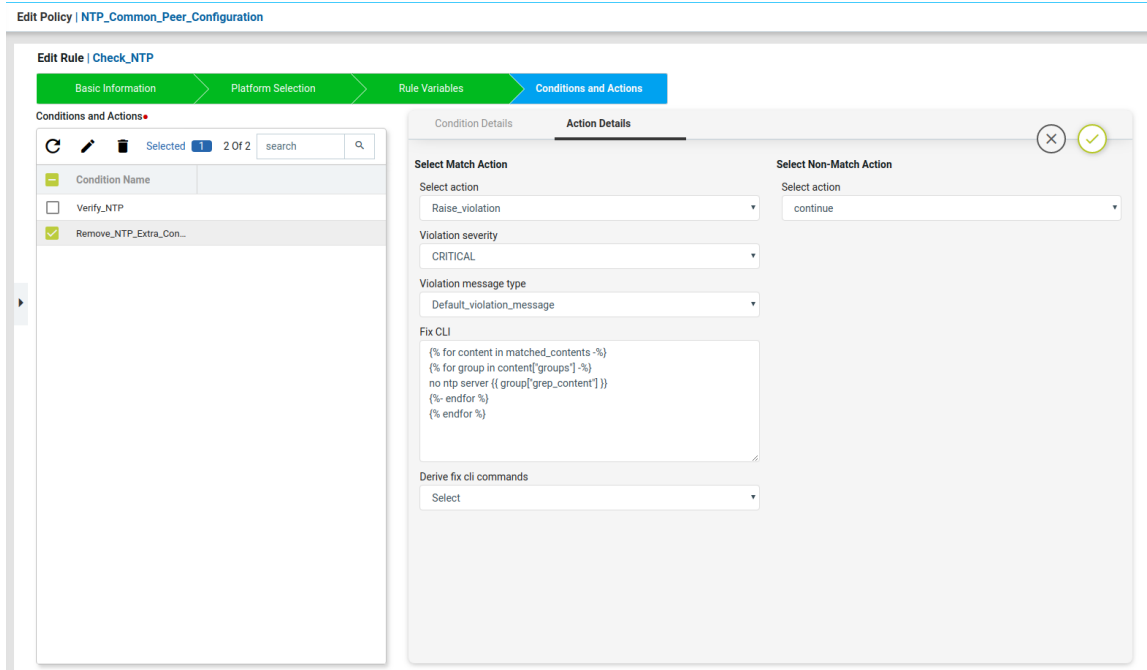
Condition2

Remove_NTP_Extra_Config condition will use the regex to match and capture the extra NTP server ip configured in the device other than the expected ip.

test-results

```
{
  "compliance-policies": {
    "highest-severity": "",
    "rule-violation-count": 0,
    "compliance-status": "compliant",
    "compliant-rules-output": {
      "violated-conditions": "",
      "device-compliance-condition-output": {
        "block-start-unmatched-content": "<![CDATA[]]>",
        "block-start-condition-search-output": "<![CDATA[{\n  \"block_start_matched_contents\": [{\n    \"groups\": [\n      {\n        \"index\": 1,\n        \"grep_content\": \"10.0.0.2\",\n        \"grep_group\": 1\n      }\n    ],\n    \"groups\": [{\n      {\n        \"index\": 1,\n        \"grep_content\": \"10.0.0.3\",\n        \"grep_group\": 1\n      }\n    }]\n  }]\n]>",
        "condition-search-output": "<![CDATA[{\n  \"matched_contents\": [{\n    \"groups\": [{\n      {\n        \"index\": 1,\n        \"grep_content\": \"10.0.0.2\",\n        \"grep_group\": 1\n      }\n    }, {\n      {\n        \"index\": 1,\n        \"grep_content\": \"10.0.0.3\",\n        \"grep_group\": 1\n      }\n    }]\n  }]\n]>",
        "total-block-count": 2,
        "aggregated-condition-output": "<![CDATA[{\n  \"condition_contents\": [{\n    \"condition_id\": null,\n    \"block_start_matched_content\": null,\n    \"block_start_unmatched_content\": null,\n    \"unmatched_content\": null,\n    \"matched_content\": null\n  }]\n]>",
        "template-substituted-content": "<![CDATA[ntp server (?!10.0.0.1)(\\d+\\.\\d+\\.\\d+\\.\\d+)]>",
        "block-unmatch-count": 0,
        "cli-match-output": "<![CDATA[ntp server 10.0.0.2\nntp server 10.0.0.3\n]]>",
        "condition-status": true,
        "unmatched-content": "<![CDATA[]]>",
        "id": "Remove_NTP_Extra_Config",
        "block-match-count": 2,
        "cli-unmatch-output": "<![CDATA[]]>"
      },
      "name": "test-condition",
      "failed-conditions": ""
    }
  }
}
```

On Match Action write a jinja2 configuration template to remove the extra ip's captured using the above test-result data structure.



Finally if different NTP servers are present on the device, for Non-Compliant device Fix CLI will show up as below

```
ntp server 10.0.0.1
no ntp server 10.0.0.2
no ntp server 10.0.0.3
```

Scenario 3: Interface configuration check

Scenario: All devices in the network should have a specific interface in no shutdown state with auto negotiation enabled. The interface block can have extra configuration commands under it but should be in no shutdown state and auto negotiation enabled.

Platform:

Cisco IOS-XE

Expected Configuration:

```
interface {{ interface_name }}
no shutdown
negotiation auto
```

Fix-CLI Configuration:

```
interface {{ interface_name }}
  no shutdown
  negotiation auto
```

This use case is an interface block configuration having rule variables. In this use case as no shutdown is generally not visible on device running config, we will check whether the interface is in shutdown or not. If shutdown it will remediate to no shutdown.

Steps:

- Navigate to Resource Manager > Config Compliance -> Policies
- Click '+' to create new Policy and provide the following information
 - Policy Name - Interfaces
 - Description
- Select the Policy and Click '+' to create new Rule
 - Rule Name - Check_Interfaces
- Navigate to Config Manager > Config Compliance -> Policies -> Rules
- Select a Rule & Provide the following information
 - Vendor - Cisco Systems
 - OS type - IOSXE
 - Device Family - ALL
 - Device Type - ALL
 - OS Version - ALL
- Now create the Rule variables for this scenario.

Edit Policy | [Interfaces](#)

Edit Rule | [Check_Interfaces](#)

Basic Information

Platform Selection

Rule Variables

Conditions and Actions

Rule Variables

↺

+

<input type="checkbox"/>	Key	Description	Default Value
<input type="checkbox"/>	interface_name		GigabitEthernet5

Conditions and Actions

Condition Details
Action Details

Condition Name
Condition Name, can contain Alphanumerics, underscore, space and hyphen ...

Sequence Number
Sequence Number controls the order of execution of the Conditions.

Scope Details
Condition scope details

Block Options
Start Expression

Condition Match Criteria
Operator
Rule-pass-criteria

Value

```
interface {{ interface_name }}
no shutdown
negotiation auto
```

Launch Test Config

In the policy we will have a jinja rule variable `interface_name`. Here `Verify_Interfaces` condition will check if the interface block config is present in the device or not and under that interface if `no shutdown` and `negotiation auto` is present.

For Scenario3 Condition Match Operator as `CONTAINS_STRING` will check whether the device configuration contains condition value or not. If device configuration contains value, the result will be Compliant, else Non-Compliant.

Test Configuration and Results

Condition Match Operator:

CONTAINS_STRING

Value

```
interface {{ interface_name }}
shutdown
negotiation auto
```

Rule Variables

Please enter the rule variables in Key: Value or Format (Ex: Key1: Value1)

```
interface_name: GigabitEthernet5
```

Test Configuration

```
interface GigabitEthernet5
description TEST
ip address 94.1.1.9 255.255.255.252
shutdown
negotiation auto
```

Test

Test Results

```
{
  "compliance-policies": {
    "highest-severity": "",
    "rule-violation-count": 0,
    "compliance-status": "compliant",
    "compliant-rules-output": {
      "violated-conditions": "",
      "device-compliance-condition-output": {
        "block-start-unmatched-content": "<[CDATA[",
        "block-start-condition-search-output": "<[CDATA[",
        "condition-search-output": "<[CDATA[\\n \\matched_contents\\": [\\n]]>",
        "total-block-count": 1,
        "aggregated-condition-output": "<[CDATA[\\n \\condition_contents\\": [\\n \\condition_id\\": null,\\n \\block_start_matched_content\\": null,\\n \\block_start_unmatched_content\\": null,\\n \\unmatched_content\\": null,\\n \\matched_content\\": null,\\n ]]>"
      }
    }
  }
}
```

Here on Non-Match Action select Continue and on Match Action add Fix cli commands to remediate on the device

Rule Variables

Conditions and Actions

Condition Details

Action Details

Select Match Action

Select action

Raise_violation

Violation severity

CRITICAL

Violation message type

Default_violation_message

Fix CLI

```
interface {{ interface_name }}
no shutdown
negotiation auto
```

Derive fix cli commands

Select

Select Non-Match Action

Select action

continue

For Non-Compliant devices Fix CLI will show up later-on as below.

```
interface GigabitEthernet5
no shutdown
negotiation auto
```

Scenario 4: Enforce VTY Session Timeouts

Scenario: All devices in the network should contain the network admin preferred VTY **session-timeout** and **exec-timeout** on all vty lines. If VTY session-timeout and exec-timeout is not configured on the device or mis-match with the network admin preferred timeouts, ATOM CLI compliance can configure the devices with the user preferred VTY timeouts on all the vty lines.

In this example we are considering the VTY session-timeout and exec-timeout as 10 sec.

Platform:

Cisco IOS-XE

Expected Configuration:

```
line vty (.*)
session-timeout 10
exec-timeout 10 10
```

Fix-CLI Configuration:

```
line vty <>
session-timeout 10
exec-timeout 10 10
```

This use case is using the regex and rule variables and uses jinja2 template for fix-cli configuration.

Steps:

- Navigate to Resource Manager > Config Compliance -> Policies
- Click '+' to create new Policy and provide the following information
 - Policy Name - Enforce_VTY_Session_Timeouts
 - Description
- Select the Policy and Click '+' to create new Rule
 - Rule Name - Check_Enforce_VTY_Session_Timeouts
- Navigate to Resource Manager > Config Compliance -> Policies -> Rules
- Select a Rule & Provide the following information
 - Vendor - Cisco Systems
 - OS type - IOSXE
 - Device Family - ALL
 - Device Type - ALL
 - OS Version - ALL
- Now create the Rule variables for this scenario.

Edit Policy | Enforce_VTY_Session_Timeouts

Edit Rule | Check_vty_session_timeouts

Basic Information > Platform Selection > Rule Variables > Conditions and Actions

Rule Variables

🔄 +

<input type="checkbox"/>	Key	Description	Default Value
<input type="checkbox"/>	exec_timeout		10
<input type="checkbox"/>	session_time...		10

Here created user defined rule variables **vty_exec_timeout** and **vty_session_timeout** with default timeout as 10. These rule variables will be used in the condition value.

The **verify_session_exec_timeouts** condition will check whether the device in the network is configured with user preferred VTY timeouts or not.

Conditions and Actions

Condition Details | Action Details

Condition Name •
Condition Name, can contain Alphanumerics, underscore, space and hyphen ...
verify_session_timeouts

Sequence Number •
Sequence Number controls the order of execution of the Conditions.
1

Scope Details
Condition scope details
Configuration

Block Options
Start Expression

Condition Match Criteria
Operator
CONTAINS_STRING
Rule-pass-criteria
All_SubBlocks

Value •
line vty (*
session-timeout {{ session_timeout }}
exec-timeout {{ exec_timeout }} 0

Launch Test Config

Here under Condition Match Criteria the Operator used was CONTAINS_STRING to check for session-timeout and exec-timeout in line vty config.

Here Rule-pass-criteria used All_SubBlocks to check the condition config in all line vty configurations of the device. If all the line vty is matching with the condition then compliant. If any of the line vty is not matching then non-compliant.

The launch Test Config will check values with the Test configuration and gives the Test Result whether compliant or not.

Test Configuration and Results

Condition Match Operator:

CONTAINS_STRING

Value

```
line vty (*)
  session-timeout {{ session_timeout }}
  exec-timeout {{ exec_timeout }} 0
```

Rule Variables

Please enter the rule variables in Key: Value or Format Ex: Key1: Value1

```
session_timeout: 10
exec_timeout: 10
```

Test Configuration

```
line vty 0 4
  session-timeout 5
  access-class ssh-permit-acl in
  exec-timeout 5 0
  privilege level 15
  transport input ssh
line vty 5 98
  session-timeout 5
  access-class ssh-permit-acl in
  exec-timeout 5 0
  privilege level 15
  transport input ssh
!
```

Test Results

```
{
  "unmatched-content": "<CDATA>[line vty 0 4\n session-timeout 5\n access-class ssh-permit-acl in\n exec-timeout 5 0\n privilege level 15\n transport input ssh\nline vty 5 98\n session-timeout 5\n access-class ssh-permit-acl in\n exec-timeout 5 0\n privilege level 15\n transport input ssh\n]",
  "cli-unmatch-output": "<CDATA>[line vty 0 4\n session-timeout 5\n access-class ssh-permit-acl in\n exec-timeout 5 0\n privilege level 15\n transport input ssh\nline vty 5 98\n session-timeout 5\n access-class ssh-permit-acl in\n exec-timeout 5 0\n privilege level 15\n transport input ssh\n]",
  "name": "test-condition",
  "failed-conditions": "",
  "block-match-count": 0,
  "verify_session_timeouts": true,
  "compliance-status": "non-compliant"
}
```

Test

Here the unmatched line vty will be captured and stored in the backend data structure. The captured data structure maximizes the view shown below.

28

test-results

```
{
  "compliance-policies": {
    "highest-severity": "",
    "rule-violation-count": 0,
    "noncompliant-rules-output": {
      "violated-conditions": "",
      "device-compliance-condition-output": {
        "block-start-unmatched-content": "<![CDATA[]]>",
        "block-start-condition-search-output": "<![CDATA[]]>",
        "condition-search-output": "<![CDATA[\\n \\\"matched_contents\\\" : [ ]\\n]]>",
        "total-block-count": 2,
        "aggregated-condition-output": "<![CDATA[\\n \\\"condition_contents\\\" : [ {\\n \\\"condition_id\\\" : null,\\n
\\\"block_start_matched_content\\\" : null,\\n \\\"block_start_unmatched_content\\\" : null,\\n \\\"unmatched_content\\\" :
null,\\n \\\"matched_content\\\" : null\\n } ]\\n]]>",
        "template-substituted-content": "<![CDATA[line vty (.*)\\n session-timeout 10\\n exec-timeout 10 0]]>",
        "block-unmatch-count": 2,
        "cli-match-output": "<![CDATA[]]>",
        "condition-status": false,
        "unmatched-content": "<![CDATA[\\n \\\"unmatched_contents\\\" : [ {\\n \\\"groups\\\" : [ {\\n \\\"index\\\" : 1,\\n
\\\"grep_content\\\" : \\\"0 4\\\",\\n \\\"grep_group\\\" : 1\\n } ]\\n }, {\\n \\\"groups\\\" : [ {\\n \\\"index\\\" : 1,\\n
\\\"grep_content\\\" : \\\"5 98\\\",\\n \\\"grep_group\\\" : 1\\n } ]\\n } ]\\n]]>",
        "id": "verify_session_timeouts",
        "block-match-count": 0,
        "cli-unmatch-output": "<![CDATA[line vty 0 4\\n session-timeout 5 \\n access-class ssh-permit-acl in\\n exec-
timeout 5 0\\n privilege level 15\\n transport input ssh\\nline vty 5 98\\n session-timeout 5 \\n access-class ssh-
permit-acl in\\n exec-timeout 5 0\\n privilege level 15\\n transport input ssh\\n]]>"
      },
      "name": "test-condition",
      "failed-conditions": ""
    },
    "compliance-status": "non-compliant"
  }
}
```

On Match action will continue and on Non-Match Action fix-cli will use the jinja2 template configuration written based on the above captured data structure.

Rule Variables

Conditions and Actions

Condition Details

Action Details

Select Match Action

Select action

continue

Select Non-Match Action

Select action

Raise_violation

Violation severity

CRITICAL

Violation message type

Default_violation_message

Fix CLI

{% for content in unmatched_contents %}
{% for group in content["groups"] %}
line vty {{ group["grep_content"] }}
session-timeout {{ session_timeout }}
exec-timeout {{ exec_timeout }} 0
exit
{% endfor %}
{% endfor %}

Derive fix cli commands

Select

The Non-compliant device fix-cli configurations derived from above jinja2 snippet will look like below.

```

line vty 0 4
  session-timeout 10
  exec-timeout 10 0
exit

line vty 5 98
  session-timeout 10
  exec-timeout 10 0
exit

```

Scenario 5: Enforce OSPF Router Id as Loopback0

Scenario: All devices in the network should contain the OSPF router-id configured with loopback0 ip address. If OSPF router-id is not configured on the device it will configure the OSPF router-id with the value of loopback0 ip address on the devices.

Platform:

Cisco IOS-XE

Expected Configuration:

```
interface Loopback0
 ip address 45.45.45.5 255.255.255.255
!
router ospf 100
 router-id 45.45.45.5
```

Fix-CLI Configuration:

```
router ospf 100
 router-id 45.45.45.5
```

This use case is using the regex and contains two conditions.

1. First condition is to capture and store loopback0 ip address. It will not have a fix-cli configuration as the intention of the condition is to capture loopback0 ip address.

Fix-cli Configuration :

<< no fix cli configuration >>

2. Second condition will check whether the OSPF router id is the same as the first condition's captured loopback0 ip address or not. if not matching then it will configure the OSPF router id with loopback0.

Fix-cli Configuration :

```
router ospf 100
 router-id 45.45.45.5
```

Steps:

- Navigate to Resource Manager > Config Compliance -> Policies
- Click '+' to create new Policy and provide the following information
 - Policy Name - Enforce_OSPF_Router_Id_as_Loopback
 - Description
- Select the Policy and Click '+' to create new Rule
 - Rule Name - Check_OSPF_Router_Id_Cisco
- Navigate to Resource Manager > Config Compliance -> Policies -> Rules
- Select a Rule & Provide the following information
 - Vendor - Cisco Systems
 - OS type - IOSXE
 - Device Family - ALL
 - Device Type - ALL
 - OS Version - ALL
- Rule variables are not required for this scenario.

[Edit Rule](#) | [Check_OSPF_Router_Id_Cisco](#)

Basic Information

Platform Selection

Rule Variables

Conditions and Actions

Conditions and Actions

↺

+

<input type="checkbox"/>	Condition Name	
<input type="checkbox"/>	Verify_Loopback0_ip	
<input type="checkbox"/>	Verify_OSPF_Router_Id_as_Loopback	

Condition1

Conditions and Actions

Condition Details

Action Details

Condition Name

Condition Name, can contain Alphanumerics, underscore, space and hyphen ...

Verify_Loopback0_ip

Sequence Number

Sequence Number controls the order of execution of the Conditions.

1

Scope Details

Condition scope details

Configuration

Block Options

Start Expression

Condition Match Criteria

Operator

CONTAINS_STRING

Rule-pass-criteria

Any_SubBlock

Value

Interface Loopback0
ip address (\d+.\d+.\d+.\d+) (\d+.\d+.\d+.\d+)

Launch Test Config

Another way of writing the above block configuration using the **Block Options** Start Expression is shown below.

The first line “interface Loopback0” can be written in the start Expression with regex symbol ^ to indicate the block starts with interface Loopback0. The remaining configuration lines can be written in value.

The screenshot displays the 'Conditions and Actions' configuration window. The 'Condition Details' tab is selected, showing the following configuration:

- Condition Name:** Verify_Loopback0
- Sequence Number:** 1
- Scope Details:** Configuration
- Block Options:** Start Expression: ^interface Loopback0
- Condition Match Criteria:** Operator: MATCHES_THE_EXPRESSION, Rule-pass-criteria: All_SubBlocks

The 'Value' field contains the regex: ip address (\d+.\d+.\d+.\d+) (\d+.\d+.\d+.\d+). A 'Launch Test Config' button is located at the bottom right of the window.

The launch test config will check the condition value with the Test configuration and will give the Test Result. Here the captured loopback0 ip address will be stored in the backend data structure as shown below.

Test Configuration and Results

Condition Match Operator:

CONTAINS_STRING

Value

interface Loopback0
ip address (ld+.ld+.ld+.ld+) (ld+.ld+.ld+.ld+)

Rule Variables

Please enter the rule variables in Key: Value => format Ex: Key1: Value1
Key1: Value1
Key2: Value2
Key3: Value3

Test Configuration

{
 interface Loopback0
 ip address 45.45.45.5 255.255.255.255
}
 interface Loopback1
 no ip address
}
 interface Loopback200
 ip address 94.1.1.1 255.255.255.255
}

Test Results

{
 "highest-severity": "",
 "rule-violation-count": 0,
 "compliance-status": "compliant",
 "compliant-rules-output": {},
 "violated-conditions": "",
 "device-compliance-condition-output": {},
 "block-start-unmatched-content": "<[CDATA]]>",
 "block-start-condition-search-output": "<[CDATA]]>",
 "condition-search-output": "<[CDATA[(\n \\'matched_contents\':[(\n \\'groups\':[(\n \\'index\':1,\n \\'grep_content\':
'45.45.45.5',\n \\'grep_group\':1\n),(\n \\'index\':2,\n \\'grep_content\':\"255.255.255.255\",
\n \\'grep_group\':
2\n)\n)\n]\n]>",
 "total-block-count": 1,
 "aggregated-condition-output": "<[CDATA[(\n \\'condition_contents\':[(\n \\'condition_id\':null,\n \\'block_start_matched_content\':null,\n \\'block_start_unmatched_content\':null,\n \\'unmatched_content\':null,\n \\'matched_content\':null\n)\n]\n]>",
 "template-substituted-content": "<[CDATA[interface Loopback0\n ip address (ld+.ld+.ld+.ld+) (ld+.ld+.ld+.ld+)]]>"

Test

The Test result in maximize view is shown below. This output will be used in condition2.

34

test-results

```
{
  "compliance-policies": {
    "highest-severity": "",
    "rule-violation-count": 0,
    "compliance-status": "compliant",
    "compliant-rules-output": {
      "violated-conditions": "",
      "device-compliance-condition-output": {
        "block-start-unmatched-content": "<![CDATA[]]>",
        "block-start-condition-search-output": "<![CDATA[]]>",
        "condition-search-output": "<![CDATA[{\n  \"matched_contents\": [{\n    \"groups\": [{\n      \"index\": 1,\n      \"grep_content\": \"45.45.45.5\\n\", \"grep_group\": 1\n    }, {\n      \"index\": 2,\n      \"grep_content\": \"255.255.255.255\\n\", \"grep_group\": 2\n    }]\n}]]>",
        "total-block-count": 1,
        "aggregated-condition-output": "<![CDATA[{\n  \"condition_contents\": [{\n    \"condition_id\": null,\n    \"block_start_matched_content\": null,\n    \"block_start_unmatched_content\": null,\n    \"unmatched_content\": null,\n    \"matched_content\": null\n  }]\n}]]>",
        "template-substituted-content": "<![CDATA[interface Loopback0\n ip address (\\d+\\.\\d+\\.\\d+\\.\\d+)\n(\\d+\\.\\d+\\.\\d+\\.\\d+)]]>",
        "block-unmatch-count": 0,
        "cli-match-output": "<![CDATA[interface Loopback0\n ip address 45.45.45.5 255.255.255.255\n]]>",
        "condition-status": true,
        "unmatched-content": "<![CDATA[{\n  \"unmatched_contents\": [ ]\n}]]>",
        "id": "Verify_Loopback0_ip",
        "block-match-count": 1,
        "cli-unmatch-output": "<![CDATA[]]>"
      },
      "name": "test-condition",
      "failed-conditions": ""
    }
  }
}
```

For Non-Match Action violation is being raised and fix-cli is having no commands as this condition is to capture the loopback0 ip.

Rule Variables
Conditions and Actions

Condition Details
Action Details

✕
✓

Select Match Action

Select action

continue

Select Non-Match Action

Select action

Raise_violation

Violation severity

CRITICAL

Violation message type

Default_violation_message

Fix CLI

Derive fix cli commands

Select

Condition2

The Verify_OSPF_Router_Id_as_Loopback condition will check whether the OSPF router id is the same as the first condition's captured loopback0 ip address or not. if not matching then in fix-cli it will configure the OSPF router id with loopback0.

Conditions and Actions

Condition Details

Action Details

Condition Name

Condition Name, can contain Alphanumerics, underscore, space and hyphen ...

Verify_OSPF_Router_Id_as_Loopback

Sequence Number

Sequence Number controls the order of execution of the Conditions.

1

Scope Details

Condition scope details

Configuration

Block Options

Start Expression

Condition Match Criteria

Operator

CONTAINS_STRING

Rule-pass-criteria

All_SubBlocks

Value

router ospf (*)
router-id {{ condition_contents[0]["matched_content"]["matched_contents"][0]["groups"][0]["grep_content"] }}

Launch Test Config

On Match Action it will continue. On Non-Match Action it will use the jinja2 template configuration in fix-cli to configure the OSPF router id with loopback0 ip.

Rule Variables

Conditions and Actions

Condition Details

Action Details

Select Match Action

Select action

continue

Select Non-Match Action

Select action

Raise_violation

Violation severity

CRITICAL

Violation message type

Default_violation_message

Fix CLI

```
{% for content in unmatched_contents %}
{% for group in content["groups"] %}
router ospf {{ group["grep_content"] }}
router-id {{ condition_contents[0]["matched_content"]
["matched_contents"][0]["groups"][0]["grep_content"] }}
exit
{% endfor %}
{% endfor %}
```

Derive fix cli commands

use-unmatched-block

The Non-compliant device fix-cli configurations from the jinja2 template configuration is given below.

Fix Configurations

```
router ospf 100
router-id 45.45.45.5
exit
```

Scenario 6: BGP TTL Hop-count

Scenario: All devices in the network should contain the network admin preferred BGP ttl-security hops. If hops is not configured on the device or mis-match with the network admin preferred ttl-security hops, ATOM CLI compliance can configure the devices with the user preferred hops.

In this example we are considering the ttl-security hops as 5.

Platform:

Cisco IOS-XE

Expected Configuration:

```
router bgp 65535
  bgp log-neighbor-changes
  neighbor 2.3.2.6 ttl-security hops 5
```

Fix-CLI Configuration:

```
router bgp 65535
  bgp log-neighbor-changes
  neighbor 2.3.2.6 ttl-security hops 5
```

This use case is using the regex and rule variables and contains two conditions.

1. First condition is to match the block. It will not have a fix-cli configuration as the intention of the condition is to match the block.

Fix-cli Configuration :

<< no fix cli configuration >>

2. Second condition will check whether the BGP ttl-security hops is in the first condition's matched block or not. if not matching in the block then it will configure the BGP hops.

Fix-cli Configuration :

```
router bgp 65535
  neighbor 2.3.2.6 ttl-security hops 5
```

Steps:

- Navigate to Resource Manager > Config Compliance -> Policies
- Click '+' to create new Policy and provide the following information
 - Policy Name - BGP_TTL_Hop_Count
 - Description
- Select the Policy and Click '+' to create new Rule
 - Rule Name - Check_BGP_TTL
- Navigate to Resource Manager > Config Compliance -> Policies -> Rules
- Select a Rule & Provide the following information
 - Vendor - Cisco Systems
 - OS type - IOSXE
 - Device Family - ALL
 - Device Type - ALL
 - OS Version - ALL
- Now create the Rule variables for this scenario.

Edit Policy | BGP_TTL_Hop_Count

Edit Rule | Check_BGP_TTL

Basic Information

Platform Selection

Rule Variables

Conditions and Actions

Rule Variables

Key	Description	Default Value
asn		65535
hops		5
neighbor		1.1.1.1

Condition1

The first line “router bgp (.*)” to be written in the start Expression with regex to indicate the block starts with router bgp. The remaining configuration lines can be written in value.

Edit Policy | BGP_TTL_Hop_Count

Edit Rule | Check_BGP_TTL

Basic Information

Platform Selection

Rule Variables

Conditions and Actions

Conditions and Actions

Condition Name

Sequence Number

Verify_BGP

1

Vnry_BGP_TTL

1

Condition Details

Condition Name

Verify_BGP

Sequence Number

1

Scope Details

Condition scope details

Configuration

Block Options

Start Expression

router bgp (.*)

Condition Match Criteria

Operator

CONTAINS_STRING

Rule-pass-criteria

Any_Subblock

Action Details

Value

bgp log-neighbor-changes

Launch Test Config

On **Match Action** execution continues to the next condition. On **Non-Match Action** it will raise a violation and continue next condition. The “Fix-CLI” for the condition was written based on the test results obtained from “Launch Test Config”.

When the start Expression is used the regex captured data will be stored in “condition_contents” of “aggregated-condition-ouput” in test results.

Test Configuration And Results

Netconf Payload To Restconf Plan

Test Configuration and Results

Condition Match Operator:

CONTAINS_STRING

Value

bgp log-neighbor-changes

Rule Variables

Preview: enter the rule variables in Key: Value <= Format Ex: Key1: Value1

asn: 65535
neighbor: 1.1.1.1
hops: 5

Test Configuration

router bgp 65535
bgp log-neighbor-changes

Test Results

```
{
  "compliance-policies": {
    "highest-severity": "",
    "rule-violation-count": 0,
    "compliance-status": "compliant",
    "compliance-rules-output": {
      "violated-conditions": ""
    },
    "device-compliance-condition-output": {
      "block-start-unmatched-content": "<[CDATA[(\n \"block_start_unmatched_contents\": [ ]\n)]>",
      "block-start-condition-search-output": "<[CDATA[(\n \"block_start_matched_contents\": [ ]\n \"groupset\": [ ]\n \"index\": 1,\n \"grep_content\": \"65535\", \n \"grep_group\": 1\n )\n ]\n)]>",
      "condition-search-output": "<[CDATA[(\n \"matched_contents\": [ ]\n)]>",
      "total-block-count": 1,
      "aggregated-condition-output": "<[CDATA[(\n \"condition_id\": \"Verify_BGP\", \n \"block_start_matched_content\": (\n \"block_start_matched_contents\": [ ]\n \"groupset\": [ ]\n \"index\": 1,\n
```

Test

test-results

```
{
  "compliance-policies": {
    "highest-severity": "",
    "rule-violation-count": 0,
    "compliance-status": "compliant",
    "compliant-rules-output": {
      "violated-conditions": "",
      "device-compliance-condition-output": {
        "block-start-unmatched-content": "<![CDATA[{\n  \"block_start_unmatched_contents\": [ ]\n}]]>",
        "block-start-condition-search-output": "<![CDATA[{\n  \"block_start_matched_contents\": [ {\n    \"groups\": [ {\n      \"index\": 1,\n      \"grep_content\": \"65535\",\n      \"grep_group\": 1\n    } ]\n  } ]\n}]]>",
        "condition-search-output": "<![CDATA[{\n  \"matched_contents\": [ ]\n}]]>",
        "total-block-count": 1,
        "aggregated-condition-output": "<![CDATA[{\n  \"condition_contents\": [ {\n    \"condition_id\": \"Verify_BGP\",\n    \"block_start_matched_content\": {\n      \"block_start_matched_contents\": [ {\n        \"groups\": [ {\n          \"index\": 1,\n          \"grep_content\": \"65535\",\n          \"grep_group\": 1\n        } ]\n      },\n      \"block_start_unmatched_content\": {\n        \"block_start_unmatched_contents\": [ ]\n      },\n      \"unmatched_content\": {\n        \"unmatched_contents\": [ ]\n      },\n      \"matched_content\": {\n        \"matched_contents\": [ ]\n      }\n    } ]\n}]]>",
        "enforcement-time": 1597311923441,
        "condition-input": "<![CDATA[bgp log-neighbor-changes]]>",
        "template-substituted-content": "<![CDATA[bgp log-neighbor-changes]]>",
        "block-unmatch-count": 0,
        "cli-match-output": "<![CDATA[router bgp 65535\n bgp log-neighbor-changes\n]]>",
        "condition-status": true,
        "unmatched-content": "<![CDATA[{\n  \"unmatched_contents\": [ ]\n}]]>",
        "id": "Verify_BGP",
        "block-match-count": 1,
        "cli-unmatch-output": "<![CDATA[]]>"
      },
    },
    "name": "test-condition",
    "failed-conditions": ""
  }
}
```



Conditions and Actions

Condition Name	Sequence Number
<input checked="" type="checkbox"/> Verify_BGP	1
<input type="checkbox"/> Verify_BGP_TTL	1

Condition Details

Select Match Action

Select action: continue

Select Non-Match Action

Select action: Raise_violation_and_continue

Violation severity: CRITICAL

Violation message type: Default_violation_message

Fix CLI:

```
bgp router (( condition_contents[0]block_star_matched_content
[block_star_matched_contents[0]group[0]group_content ]))
bgp log-neighbor-changes
```

Derive fix cli commands: use-unmatched-block

Condition2

The Verify_BGP_TTL condition will check whether the router bgp block config matched in the previous condition has the ttl-security hops or not. if not matching then in fix-cli it will configure the ttl-security hops.

This condition uses the **condition scope details** as **Previously_Matched_Blocks** to check on previous condition matched block.

Edit Rule | Check_BGP_TTL

Conditions and Actions

Condition Name	Sequence Number
<input type="checkbox"/> Verify_BGP	1
<input checked="" type="checkbox"/> Verify_BGP_TTL	1

Condition Details

Condition Name: Verify_BGP_TTL

Sequence Number: 1

Scope Details

Condition scope details: Previously_Matched_Blocks

Condition Match Criteria

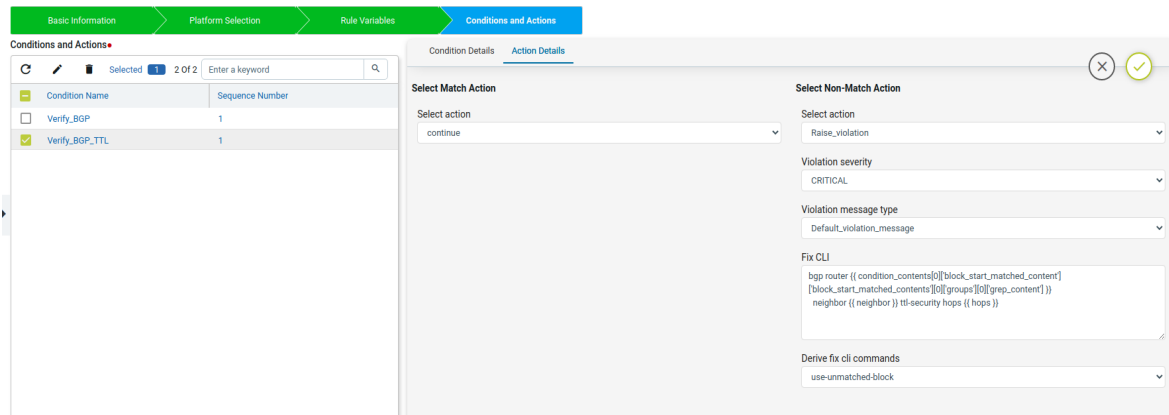
Operator: CONTAINS_STRING

Rule-pass-criteria: Any_SubBlock

Value

```
neighbor (( neighbor )) ttl-security hops (( hops ))
```

[Launch Test Config](#)



YANG Compliance

Note: In order to use Yang Compliance make sure that the config-snapshot is provided in the Credential profile, which lets ATOM to parse the configuration and store it. For more information on Credential profile please refer to credential profile section in ATOM User guide.

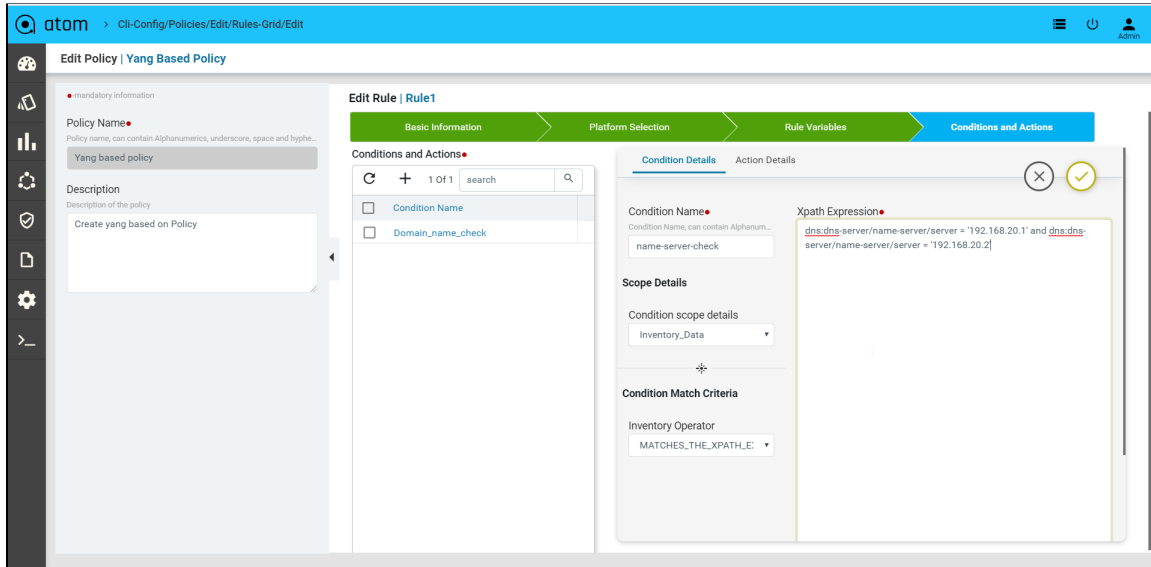
For Yang based Configuration Compliance, make sure to select the option of **Inventory_Data** for **Condition scope** during Compliance **Policy** creation. This gives two ways of defining the **Condition Match Criteria**

- Xpath Expressions
- XML Template Payload

Policy creation with Xpath Expressions

- Within **Condition Match Criteria** select “Matches_the_Xpath_Expression” / “Doesn’t_Matches_the_Xpath_Expression” option for **Inventory Operator** field
- The Fix Mutation Payload is in Netconf xml RPC format written using the XML template details for the yang parsed entities.

Navigate to Resource Manager > Config Compliance > Policy > + (Add Policies)



Few examples

Scenario 7: IP Domain Name

In this example we are looking for the domain name as **anutacorp.com** across all devices in the lab using X-path expression.

Xpath Expression:

```
Cisco-IOS-XR-native:native/ip/domain/name='anutacorp.com'
```

Fix Mutation Payload:

```
<config>
  <devices xmlns="http://anutanetworks.com/controller">
    <device>
      <id>{{ inputDeviceId }}</id>
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <ip>
          <domain nc:operation='create'>
            <name>anutacorp.com</name>
          </domain>
        </ip>
      </native>
    </device>
  </devices>
</config>
```

```
</native>
</device>
</devices>
</config>
```

Defining Xpath Expression

Edit Policy | Yang_IP_Domain_Name

Edit Rule | Check_Yang_IP_Domain_Name

Basic Information > Platform Selection > Rule Variables > Conditions and Actions

Conditions and Actions

Condition Name	Sequence Number
Verify_Yang_IP_Domain_Name	1

Condition Details

Condition Name: Verify_Yang_IP_Domain_Name

Sequence Number: 1

Scope Details

Condition scope details: Inventory_Data

Condition Match Criteria

Inventory Operator: MATCHES_THE_XPATH_EXPRESSION

Xpath Expression

Xpath Expression Ex: `property[Name='value', starts-with(property[Name='value'], contains(property[Name='value']`

Cisco-IOS-XE-native:ip/domain/name='anutacorp.com'

Launch Test Config

Defining Fix Payload

Edit Policy | Yang_IP_Domain_Name

Edit Rule | Check_Yang_IP_Domain_Name

Basic Information > Platform Selection > Rule Variables > Conditions and Actions

Conditions and Actions

Condition Name	Sequence Number
Verify_Yang_IP_Domain_Name	1

Action Details

Select Match Action

Select action: continue

Select Non-Match Action

Select action: Raise_violation

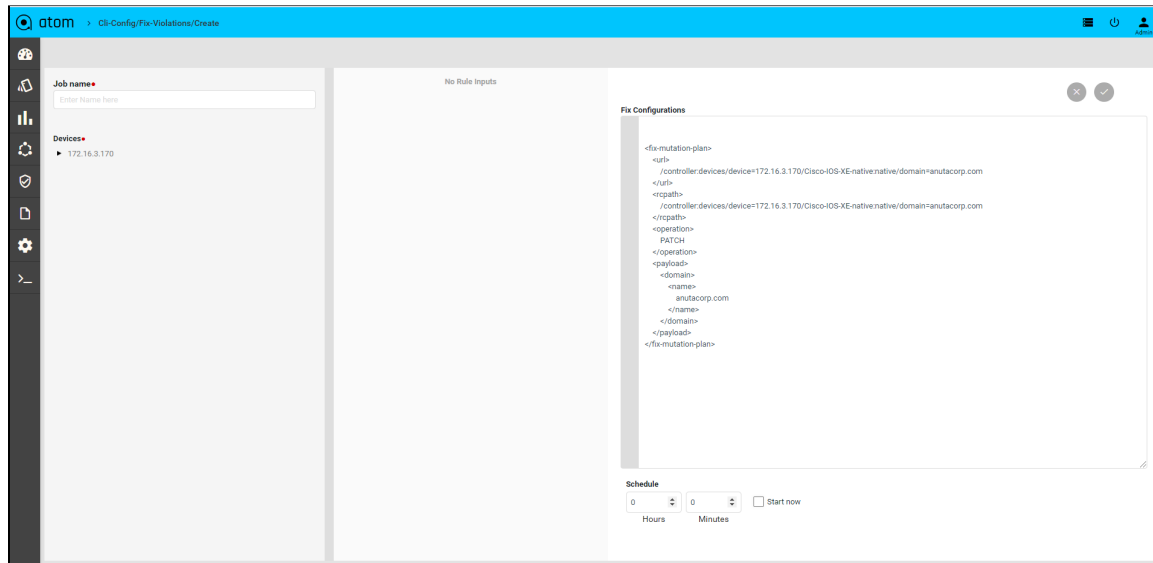
Violation severity: CRITICAL

Violation message type: Default_violation_message

Fix Mutation Payload

```
<config>
  <devices xmlns="http://anutanetworks.com/controller">
    <device>
      <id-{ {inputDeviceId} }</id>
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <ip>
          <domain nc:operation='create'>
            <name=anutacorp.com</name>
          </domain>
        </ip>
      </native>
    </device>
  </devices>
</config>
```

Fix Configuration Display in Remediation



Scenario 8: IP Name-server check

Xpath Expression:

Cisco-IOS-XE-native:native/ip/name-server/no-vrf='192.168.20.1' and
Cisco-IOS-XE-native:native/ip/name-server/no-vrf='192.168.20.2'

Fix Mutation Payload:

```
<config>
  <devices xmlns="http://anutanetworks.com/controller">
    <device>
      <id>{{ inputDeviceId }}</id>
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <ip>
          <name-server nc:operation='create'>
            <no-vrf>192.168.20.1</no-vrf>
            <no-vrf>192.168.20.2</no-vrf>
          </name-server>
        </ip>
      </native>
    </device>
  </devices>
</config>
```

Defining Xpath Expression

The screenshot shows the ATOM configuration interface for editing a policy named 'YANG_IP_Name_Server'. The 'Conditions and Actions' tab is active, and the 'Verify_IP_Name_Server' condition is selected. The 'Xpath Expression' field is defined as: `xpath Expression Ex: propertyName='value', starts-with(propertyName,value), contains(propertyName,value)`. The 'Condition Name' is 'Verify_IP_Name_Server' and the 'Sequence Number' is 1. The 'Scope Details' are set to 'Inventory_Data' and the 'Condition Match Criteria' is 'MATCHES_THE_XPATH_EXPRESSION'. A 'Launch Test Config' button is visible at the bottom right.

Defining Fix Payload

The screenshot shows the ATOM configuration interface for editing a policy named 'YANG_IP_Name_Server'. The 'Conditions and Actions' tab is active, and the 'Verify_IP_Name_Server' condition is selected. The 'Select Match Action' is set to 'continue'. The 'Select Non-Match Action' is set to 'Raise_violation'. The 'Violation severity' is 'CRITICAL' and the 'Violation message type' is 'Default_violation_message'. The 'Fix Mutation Payload' is defined as:

```
<config>
<devices xmlns="http://anutanetworks.com/controller">
  <device>
    <id>{ {inputDeviceId} }</id>
    <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
      <ip>
        <name-server nc:operation="create">
          <no-vrf>192.168.20.1</no-vrf>
          <no-vrf>192.168.20.2</no-vrf>
        </name-server>
      </ip>
    </native>
  </device>
</devices>
</config>
```

Scenario 9 : NTP server Check

Xpath Expression:

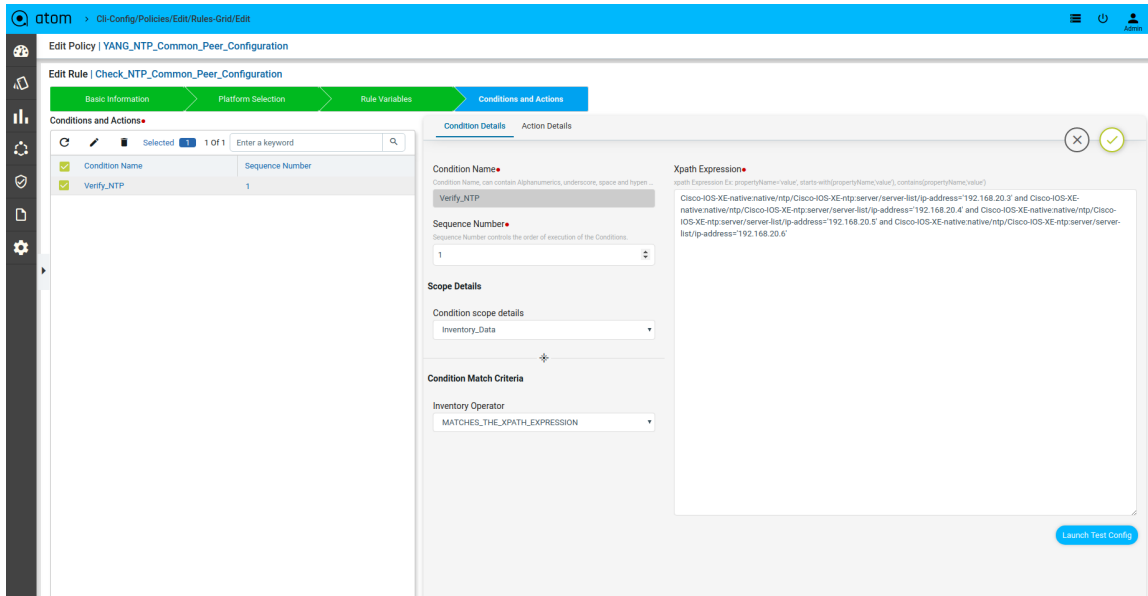
Cisco-IOS-XE-native:native/ntp/Cisco-IOS-XE-ntp:server/server-list/ip-address='192.168.20.3' and
Cisco-IOS-XE-native:native/ntp/Cisco-IOS-XE-ntp:server/server-list/ip-address='192.168.20.4' and

Cisco-IOS-XE-native:native/ntp/Cisco-IOS-XE-ntp:server/server-list/ip-address='192.168.20.5' and
Cisco-IOS-XE-native:native/ntp/Cisco-IOS-XE-ntp:server/server-list/ip-address='192.168.20.6'

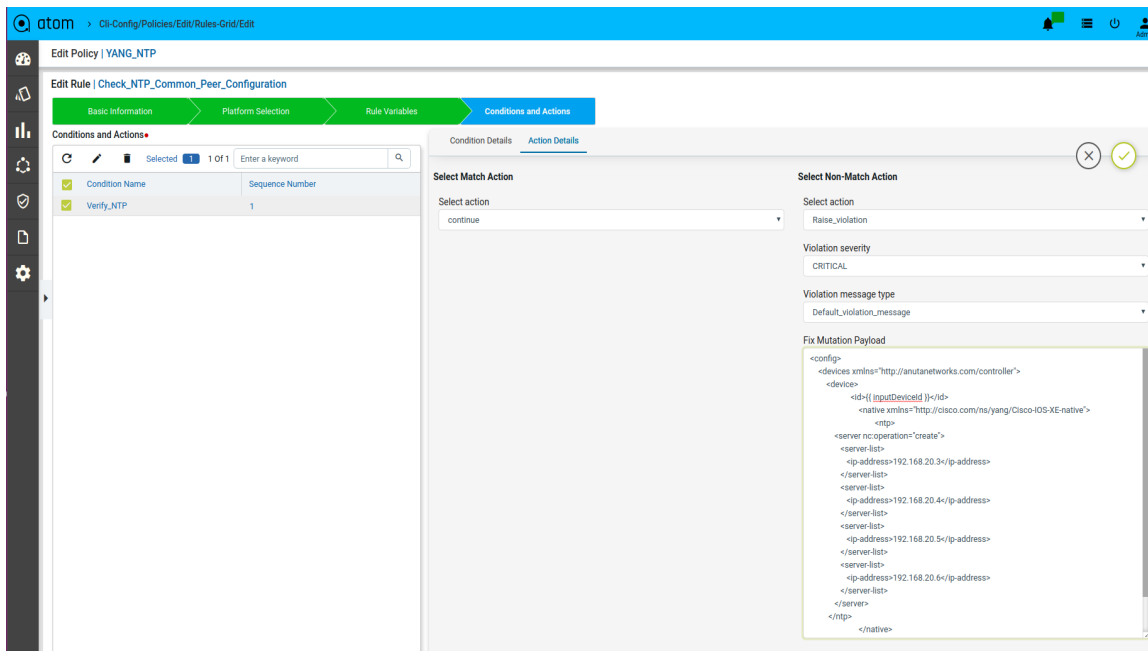
Fix Mutation Payload:

```
<config>
  <devices xmlns="http://anutanetworks.com/controller">
    <device>
      <id>{{ inputDeviceId }}</id>
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <ntp>
          <server nc:operation="create">
            <server-list>
              <ip-address>192.168.20.3</ip-address>
            </server-list>
            <server-list>
              <ip-address>192.168.20.4</ip-address>
            </server-list>
            <server-list>
              <ip-address>192.168.20.5</ip-address>
            </server-list>
            <server-list>
              <ip-address>192.168.20.6</ip-address>
            </server-list>
          </server>
        </ntp>
      </native>
    </device>
  </devices>
</config>
```

Defining Xpath Expression



Defining Fix Payload



Scenario 10 : Interface Check with rule_variable

Xpath Expression:

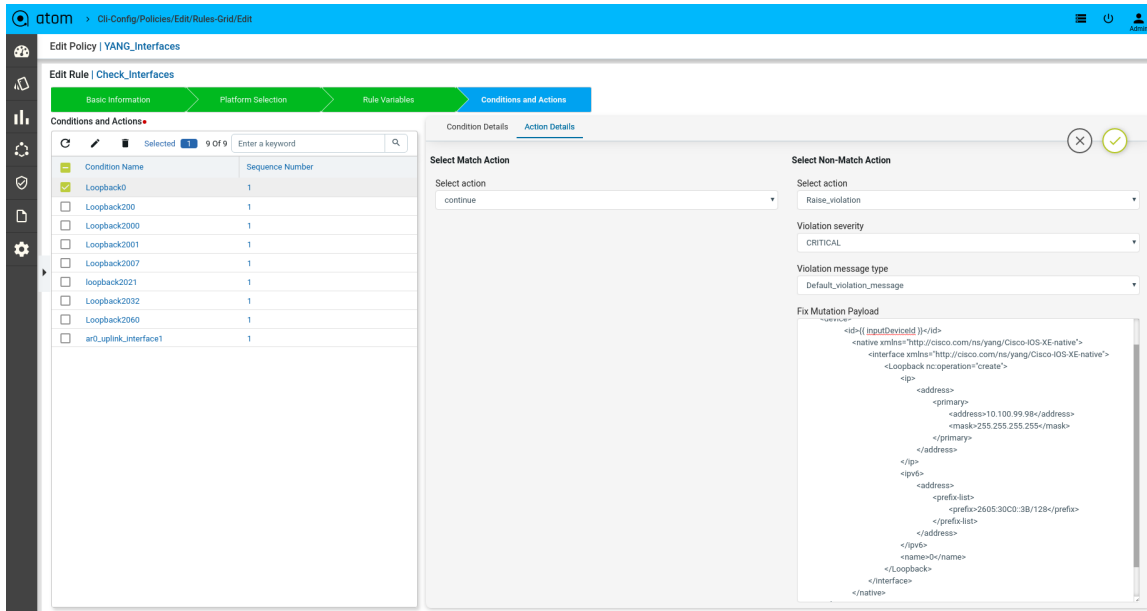
Cisco-IOS-XE-native:native/interface/Loopback/name='0' and
Cisco-IOS-XE-native:native/interface/Loopback[name=0]/ip/address/primary/address=
'{{ lo0_ipv4addr }}' and

Cisco-IOS-XE-native:native/interface/Loopback[name=0]/ip/address/primary/mask='255.255.255.255' and
 Cisco-IOS-XE-native:native/interface/Loopback[name=0]/ipv6/address/prefix-list/prefix='{{ lo0_ipv6addr }}'

Fix Mutation Payload:

```
<config>
  <devices xmlns="http://anutanetworks.com/controller">
    <device>
      <id>{{ inputDeviceId }}</id>
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <interface>
          <Loopback nc:operation="create">
            <ip>
              <address>
                <primary>
                  <address>10.100.99.98</address>
                  <mask>255.255.255.255</mask>
                </primary>
              </address>
            </ip>
            <ipv6>
              <address>
                <prefix-list>
                  <prefix>2605:30C0::3B/128</prefix>
                </prefix-list>
              </address>
            </ipv6>
            <name>0</name>
          </Loopback>
        </interface>
      </native>
    </device>
  </devices>
</config>
```

Defining Rule Variables



Scenario 11 : VRF Check with rule_variable

Xpath Expression:

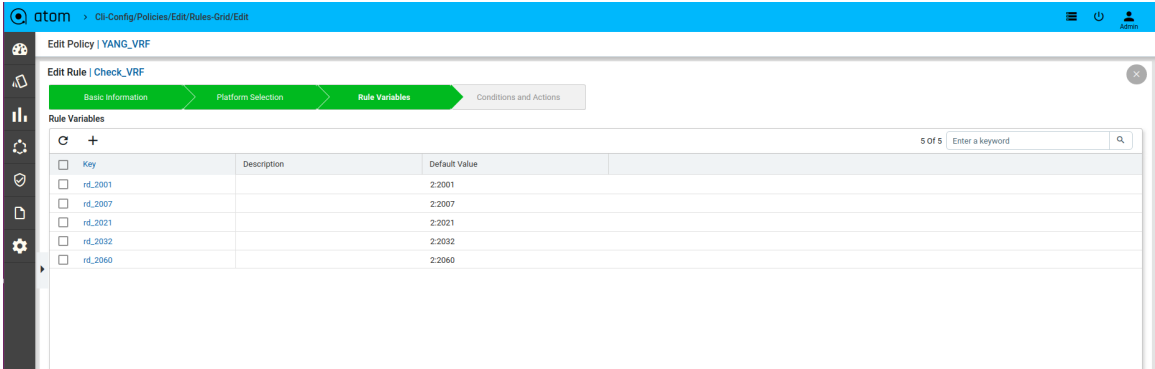
Cisco-IOS-XE-native:native/vrf/definition/name='LON2001' and
 Cisco-IOS-XE-native:native/vrf/definition/rd='{{ rd_2001 }}' and
 Cisco-IOS-XE-native:native/vrf/definition/address-family/ipv4/mdt/default/address=23
 9.232.0.1' and
 Cisco-IOS-XE-native:native/vrf/definition/address-family/ipv4/mdt/data/multicast/add
 ress='239.232.1.0' and
 Cisco-IOS-XE-native:native/vrf/definition/address-family/ipv4/mdt/data/multicast/wil
 dcard='0.0.0.255' and
 Cisco-IOS-XE-native:native/vrf/definition/address-family/ipv4/route-target/export/asn
 -ip='2:2001' and
 Cisco-IOS-XE-native:native/vrf/definition/address-family/ipv4/route-target/import/asn
 -ip='2:2001'

Fix Mutation Payload:

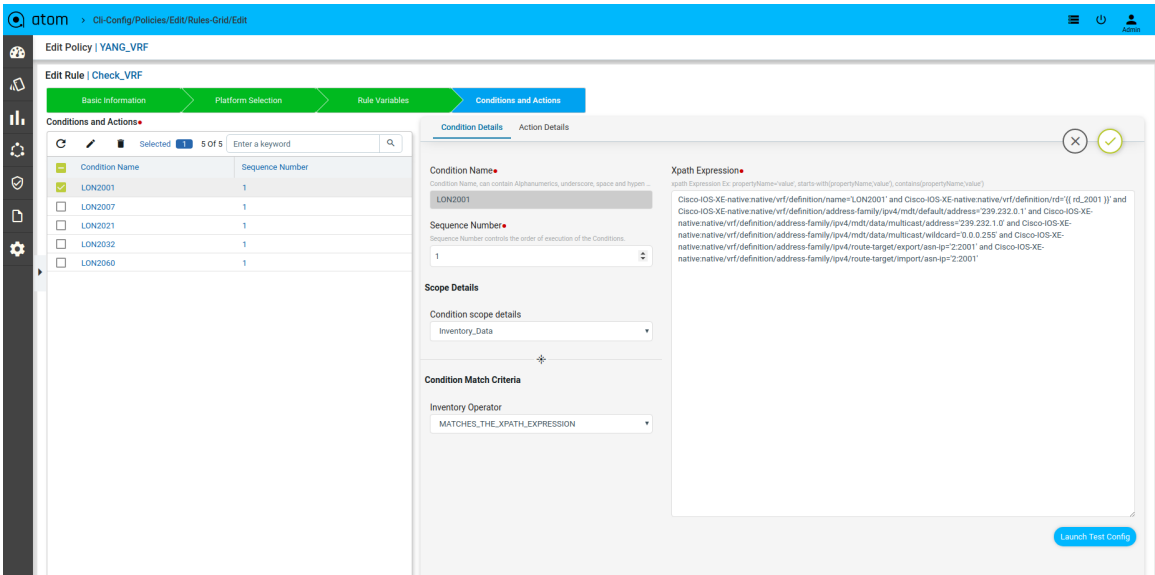
```
<config>
  <devices xmlns="http://anutanetworks.com/controller">
    <device>
      <id>{{ inputDeviceId }}</id>
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <vrf>
```

```
<definition nc:operation="create">
  <rd>2:2001</rd>
  <name>LON2001</name>
  <address-family>
    <ipv4>
      <route-target>
        <export>
          <asn-ip>2:2001</asn-ip>
        </export>
        <import>
          <asn-ip>2:2001</asn-ip>
        </import>
      </route-target>
      <mdt>
        <default>
          <address>239.232.0.1</address>
        </default>
        <data>
          <multicast>
            <address>239.232.1.0</address>
            <wildcard>0.0.0.255</wildcard>
          </multicast>
        </data>
      </mdt>
    </ipv4>
  </address-family>
</definition>
</vrf>
</native>
</device>
</devices>
</config>
```

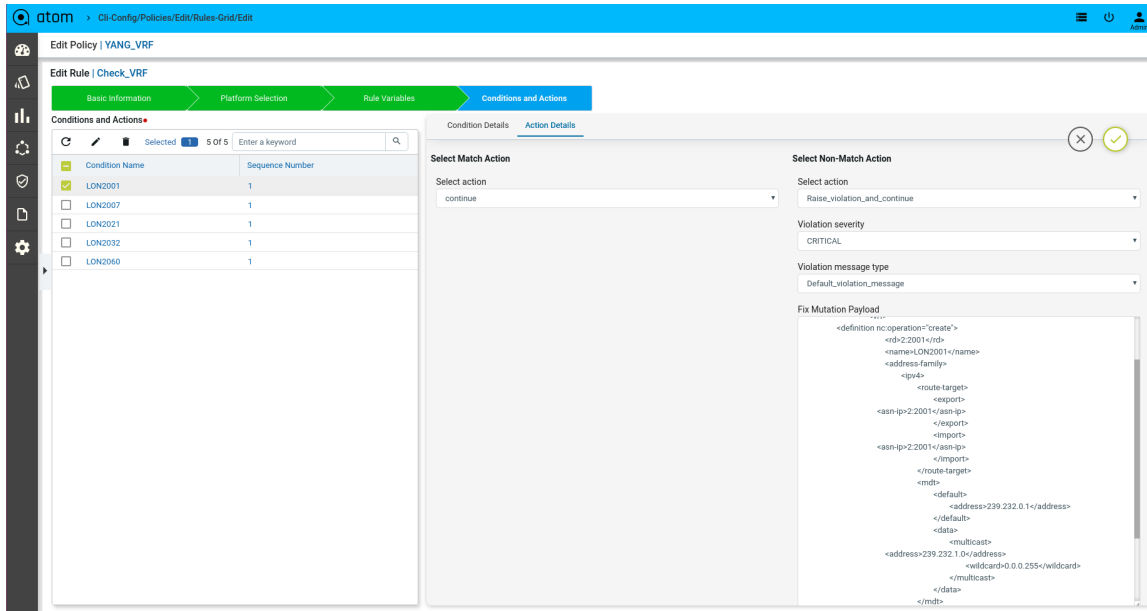
Defining Rule Variables



Defining Xpath Expression



Defining Fix Payload



- **Snmp-string with rule_variable :**
basicDeviceConfigs:snmp/snmp-community-list/snmp-string = "{{ community }}"
- **Logical A|B :** starts-with(vendor-string,'Cisco') and contains(device-family-string,'Cisco 800')
- **Logical A&B :** starts-with(vendor-string,'Cisco') or contains(device-family-string,'Cisco 800')
- **Logical A&(B|C) :** contains(vendor-string,'Cisco Systems') and (contains(device-family-string,'Cisco 800') or contains(device-family-string,'Cisco CSR 1000V'))
- **Logical A&(B|(C&D)) :**
contains(interface:interfaces/interface/if-name,'GigabitEthernet1') and (contains(os-version,'15.6(1)S') or (contains(vendor-string,'Cisco Systems') and contains(device-family-string,'Cisco CSR 1000V')))
- **Logical not(A&B) :**
not(contains(basicDeviceConfigs:local-credentials/local-credential/name , 'admin') and contains(basicDeviceConfigs:local-credentials/local-credential/name , 'cisco'))

How to derive the X-path expressions

There can be two ways by which you can derive the X-path expressions

- Navigate to the Device profile page to get the X-path Expression Details for the yang parsed entities

Resource Manager → Devices → Select a Device → Configuration → Config Data → Entities → Select Entity

For Example: If we want to write xpath expression for VRF name to match as “**anuta**”, then below is how condition needs to be written

`l3features:vrfs/vrf/name = 'anuta'`

`l3features:vrfs/vrf` : **This is x-path derived based on model under device**

`name` : **Attribute of vrf name.**

- Navigate to **Schema Browser** to see all yang models under path `/controller:devices/device`

Policy creation with XML Template Payload

- Within **Condition Match Criteria** select “Matches_the_template_payload”
/”Doesn’t_matches_the_template_payload” option for **Inventory Operator** field
- The Fix Mutation Payload is a Jinja2 template configuration in Netconf xml RPC format written using the unmatched content from the test results tab.

Navigate to Resource Manager > Config Compliance > Policy > + (Add Policies)

Few examples

Scenario 12 : IP Domain name check

Template Payload:

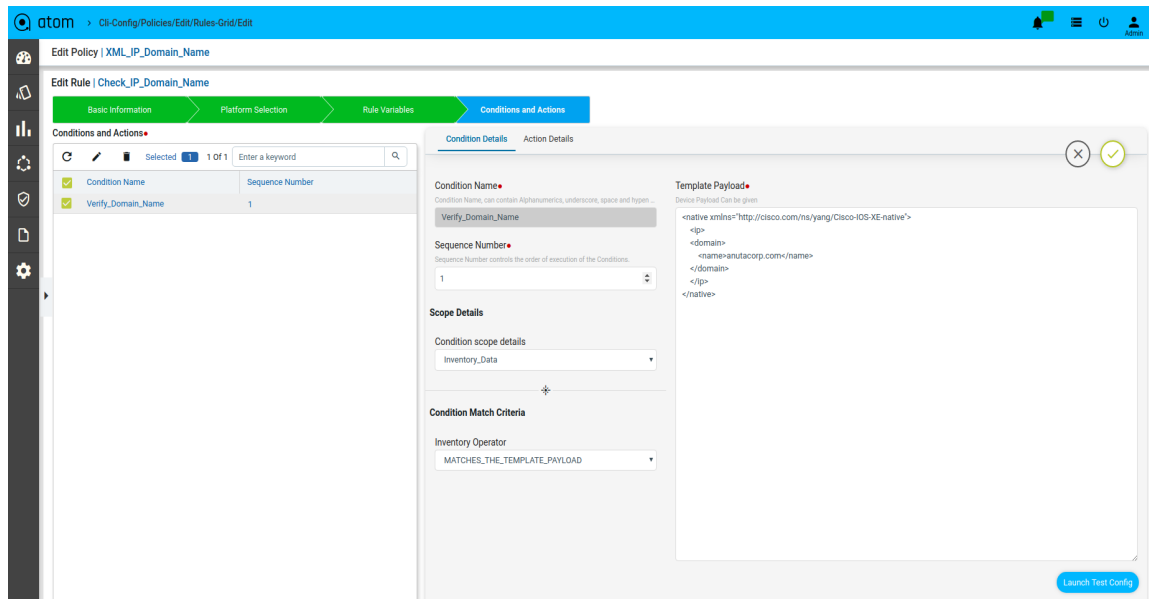
```
<native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
  <ip>
    <domain>
      <name>net.disney.com</name>
    </domain>
  </ip>
</native>
```

Fix Mutation Payload :

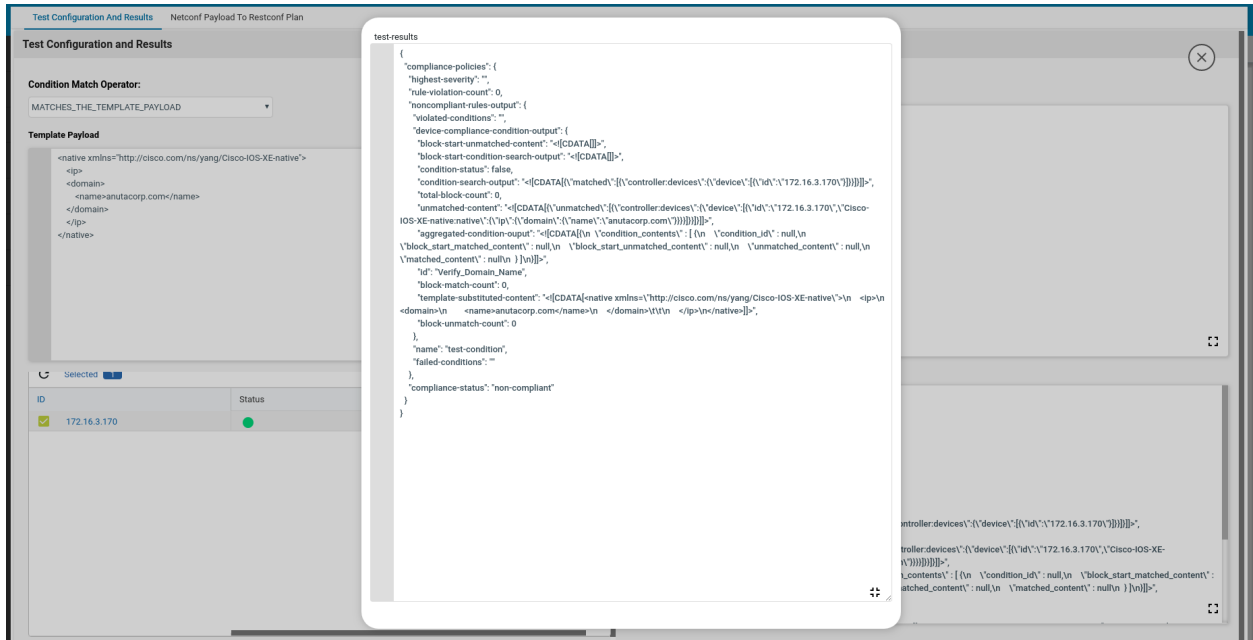
```
<config>
  <devices xmlns="http://anutanetworks.com/controller">
    {% for content in unmatched -%}
    {% for device in content["controller:devices"]["device"] -%}
    <device>
```

```
<id>{{ device["id"] }}</id>
<native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
  <ip>
    <domain nc:operation='create'>
      <name>{{ device['Cisco-IOS-XE-native:native']['ip']['domain']['name'] }}</name>
    </domain>
  </ip>
</native>
</device>
{% - endfor %}
{% - endfor %}
</devices>
</config>
```

Defining Template Payload

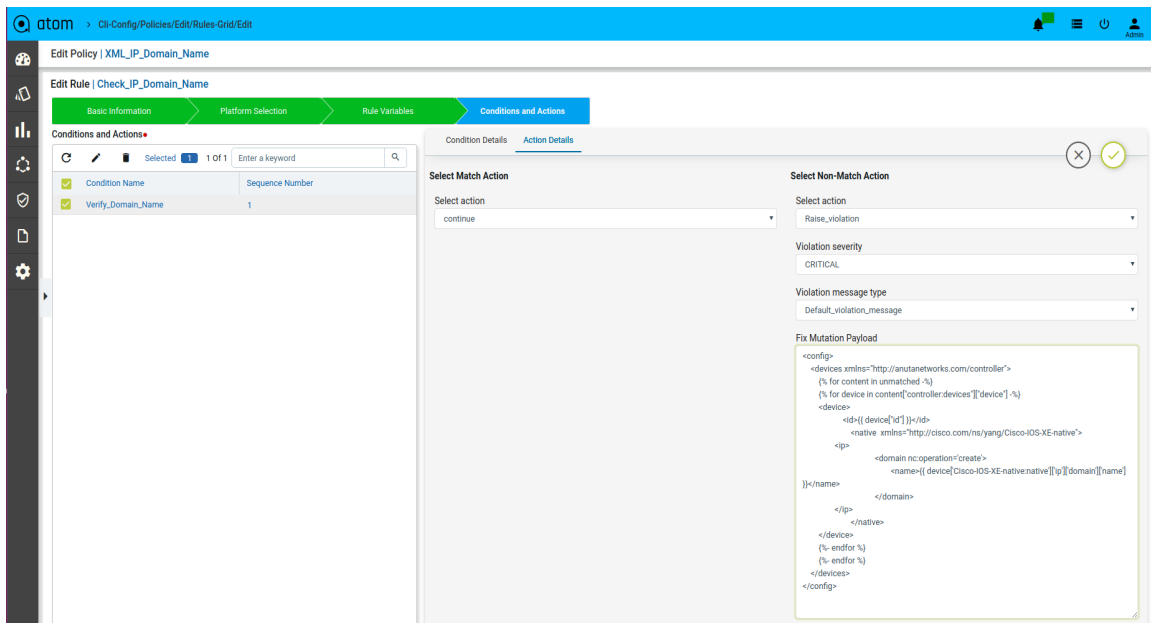


Here the matched and unmatched data will be stored in the backend data structure which is shown in the Test Results tab. The matched data will be stored in the condition-search-output. The unmatched data will be stored in unmatched-content.

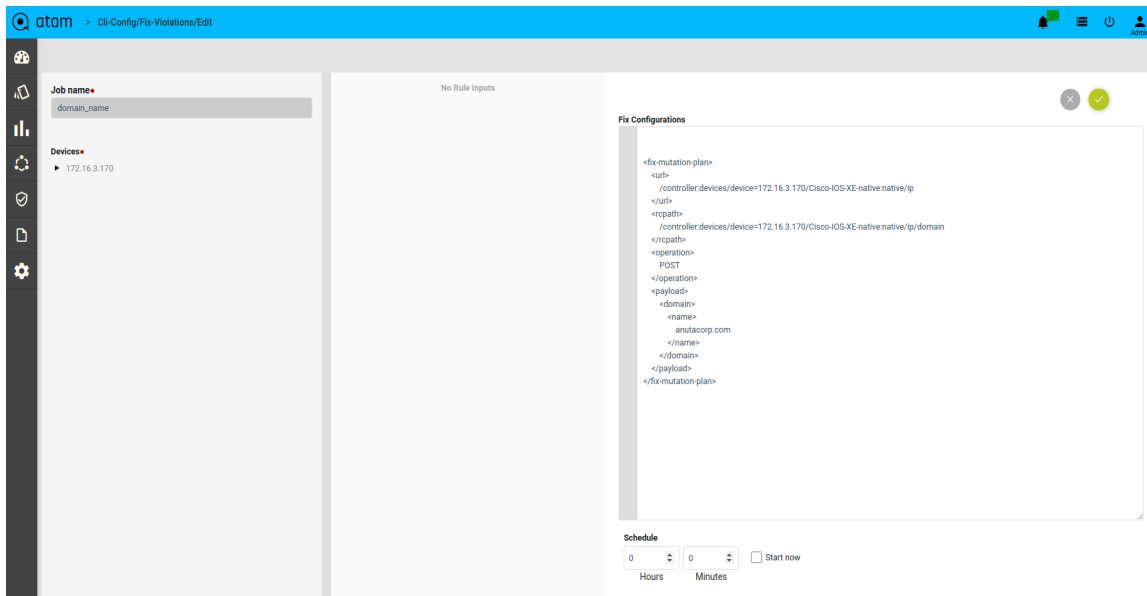


The Fix Mutation Payload is a Jinja2 template configuration in Netconf xml RPC format written using the unmatched content from the test results tab.

Defining Fix Payload



Fix Configuration Display in Remediation



Scenario 13 : IP Name Server check

Template Payload:

```
<native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
  <ip>
    <name-server>
      <no-vrf>192.168.20.1</no-vrf>
      <no-vrf>192.168.20.2</no-vrf>
    </name-server>
  </ip>
</native>
```

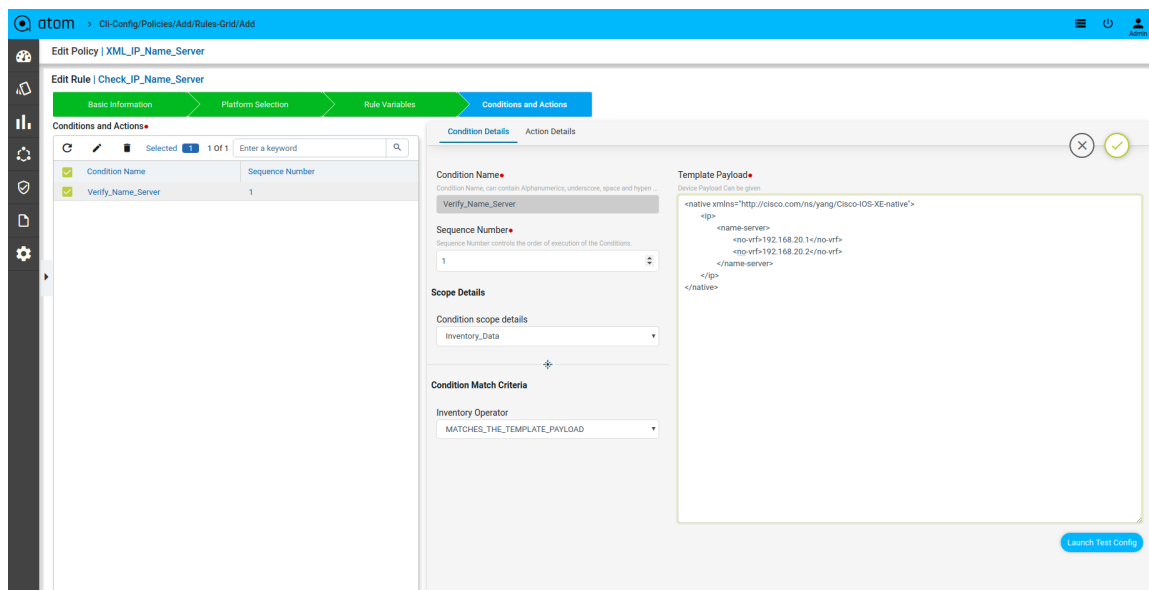
Fix Mutation Payload :

```
<config>
  <devices xmlns="http://anutanetworks.com/controller">
    {% for content in unmatched -%}
    {% for device in content["controller:devices"]["device"] -%}
    <device>
      <id>{{ device["id"] }}</id>
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <ip>
          <name-server nc:operation='create'>
```

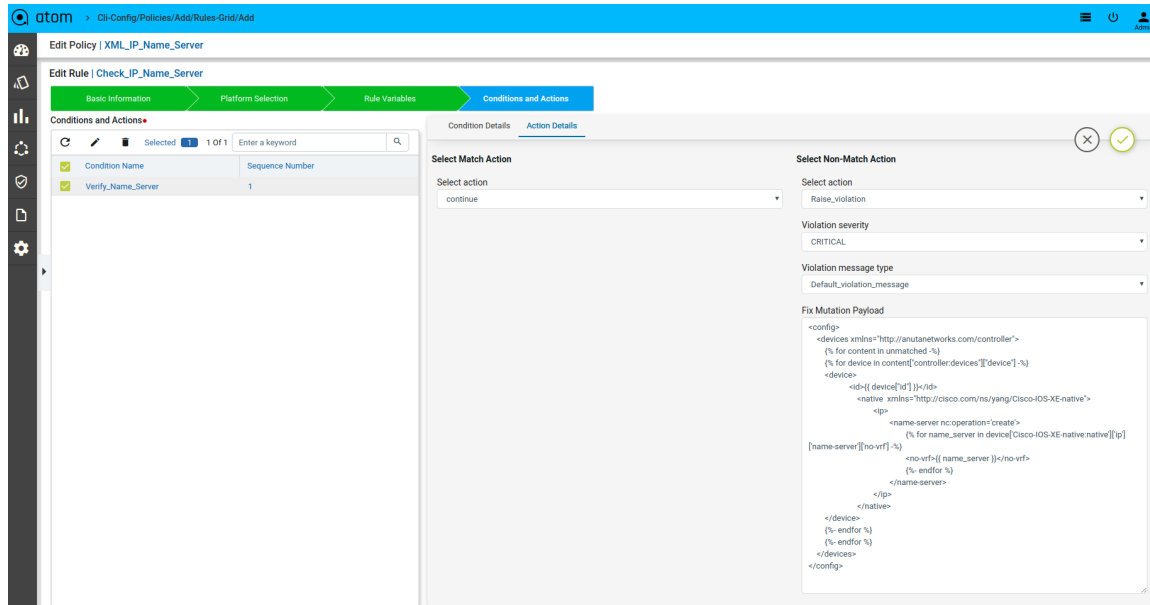
```

        {% for name_server in
device['Cisco-IOS-XE-native:native']['ip']['name-server']['no-vrf'] -%}
        <no-vrf>{{ name_server }}</no-vrf>
        {%- endfor %}
    </name-server>
</ip>
</native>
</device>
{%%- endfor %}
{%%- endfor %}
</devices>
</config>

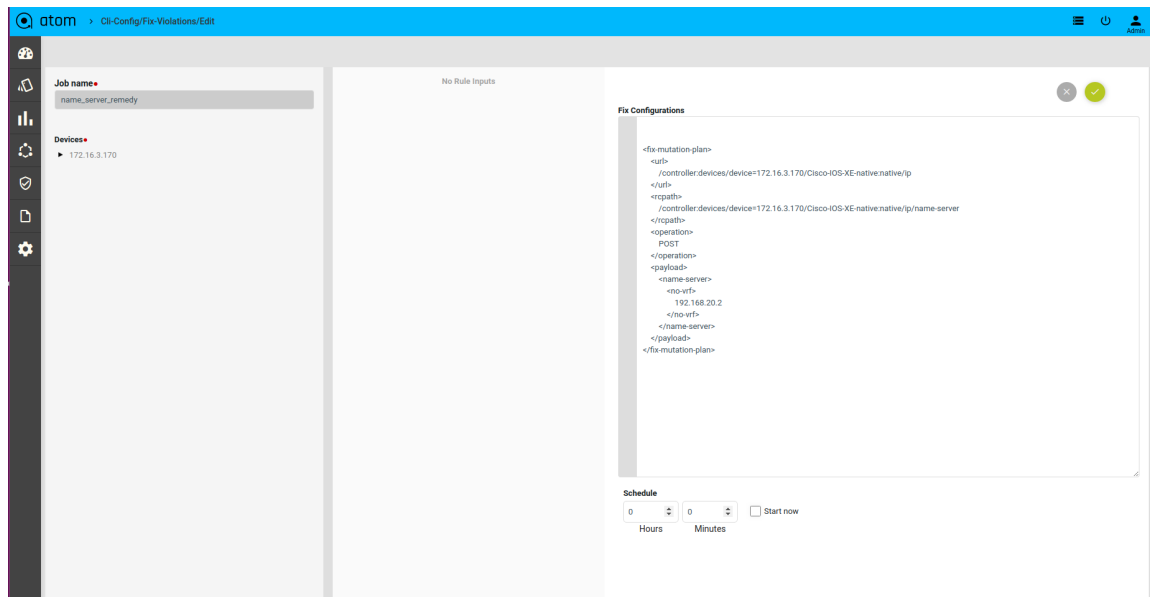
```



Defining Fix Payload



Fix Configuration Display in Remediation



Scenario 14 : Interface check

Template Payload:

```
<native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
  <interface>
    <Loopback>
      <ip>
```

```

    <address>
      <primary>
        <address>10.100.99.98</address>
        <mask>255.255.255.255</mask>
      </primary>
    </address>
  </ip>
  <ipv6>
    <address>
      <prefix-list>
        <prefix>2605:30C0::3B/128</prefix>
      </prefix-list>
    </address>
  </ipv6>
  <name>0</name>
</Loopback>
</interface>
</native>

```

Fix Mutation Payload :

```

<config>
  <devices xmlns="http://anutanetworks.com/controller">
    {% for content in unmatched -%}
    {% for device in content["controller:devices"]["device"] -%}
    <device>
      <id>{{ device["id"] }}</id>
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <interface>
          {% for loopback in
device['Cisco-IOS-XE-native:native']['interface']['Loopback'] -%}
          <Loopback nc:operation="create">
            {% if loopback['ip'] -%}
            <ip>
              <address>
                <primary>
                  <address>{{ loopback['ip']['address']['primary']['address']
}}</address>
                  <mask>{{ loopback['ip']['address']['primary']['mask'] }}</mask>
                </primary>

```

```

</address>
</ip>
{% - endif %}
{% if loopback['ipv6'] -%}
<ipv6>
<address>
    <prefix-list>
    {% for prefix in loopback['ipv6']['address']['prefix-list'] -%}
    <prefix>{{ prefix['prefix'] }}</prefix>
    {% - endfor %}
</prefix-list>
</address>
</ipv6>
{% - endif %}
<name>{{ loopback['name'] }}</name>
</Loopback>
{% - endfor %}
</interface>
</native>
</device>
{% - endfor %}
{% - endfor %}

</devices>
</config>

```

Defining Template Payload

The screenshot shows the ATOM configuration interface. The top navigation bar includes 'atom' and 'CLI Config/Policies/Edit/Rules-Grid/Edit'. The main header is 'Edit Rule | Check_Interfaces'. Below this, there are tabs for 'Basic Information', 'Platform Selection', 'Rule Variables', and 'Conditions and Actions'. The 'Conditions and Actions' tab is selected, showing a table of conditions and a 'Template Payload' section.

Condition Name	Sequence Number
<input checked="" type="checkbox"/> Loopback0	1
<input type="checkbox"/> Loopback200	1
<input type="checkbox"/> Loopback2000	1
<input type="checkbox"/> Loopback2001	1
<input type="checkbox"/> Loopback2007	1
<input type="checkbox"/> Loopback2021	1
<input type="checkbox"/> Loopback2032	1
<input type="checkbox"/> Loopback2060	1
<input type="checkbox"/> ar0_uplink_interface1	1

Condition Details

Condition Name: Loopback0

Sequence Number: 1

Scope Details

Condition scope details: Inventory_Data

Condition Match Criteria

Inventory Operator: MATCHES_THE_TEMPLATE_PAYLOAD

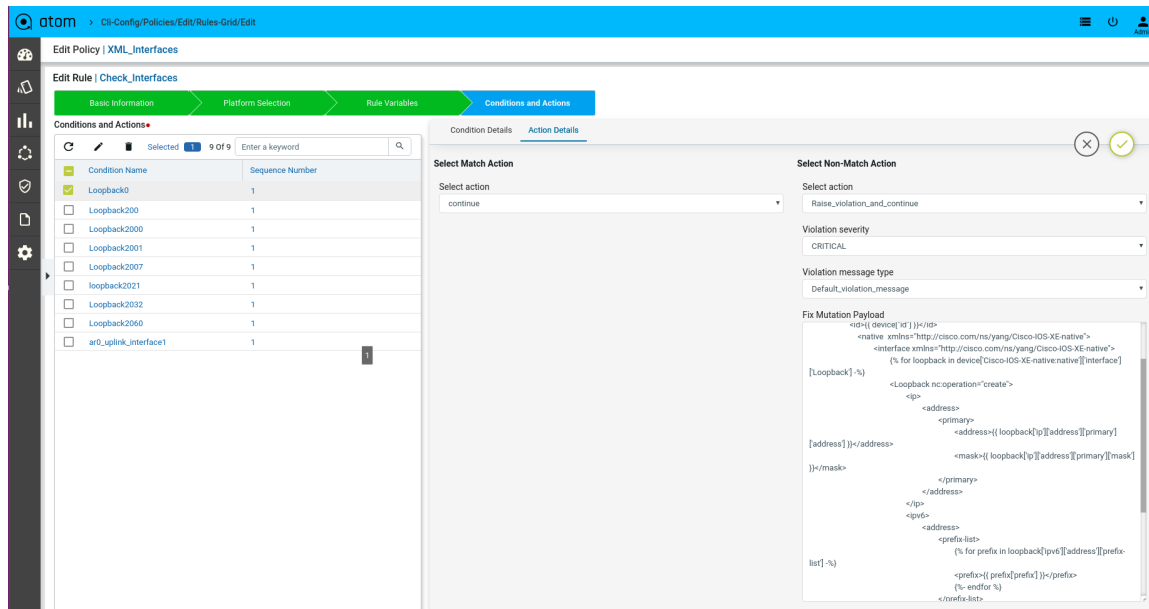
Template Payload

```

<native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
  <interface>
    <loopback>
      <ip>
        <address>
          <primary>
            <address>10.100.99.98</address>
            <mask>255.255.255.255</mask>
          </primary>
          <address>
            <ip>
              <address>
                <prefix-list>
                  <prefix>2605:30C0:3B:128::/prefix>
                </prefix-list>
              </address>
            </ip>
          </address>
          <name>0</name>
        </loopback>
      </interface>
    </native>
  
```

Launch Test Config

Defining Template Payload



Scenario 15 : VRF check

Template Payload:

```
<native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
  <vrf>
    <definition>
      <address-family>
        <ipv4>
          <mdt>
            <default>
              <address>239.232.0.1</address>
            </default>
            <data>
              <multicast>
                <address>239.232.1.0</address>
                <wildcard>0.0.0.255</wildcard>
              </multicast>
            </data>
          </mdt>
          <route-target>
            <export>
              <asn-ip>2:2001</asn-ip>
            </export>
          </route-target>
        </ipv4>
      </address-family>
    </definition>
  </vrf>
</native>
```

```

        <asn-ip>2:2001</asn-ip>
    </import>
</route-target>
</ipv4>
</address-family>
<name>LON2001</name>
<rd>2:2001</rd>
</definition>
</vrf>
</native>

```

Fix Mutation Payload :

```

<config>
  <devices xmlns="http://anutanetworks.com/controller">
    {% for content in unmatched -%}
    {% for device in content["controller:devices"]["device"] -%}
    <device>
      <id>{{ device["id"] }}</id>
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <vrf>
          {% for vrf_def in device['Cisco-IOS-XE-native:native']['vrf']['definition']
-%}

          {% if vrf_def['name'] == 'LON2001' -%}
          <definition nc:operation="create">
            {% if vrf_def['rd'] -%}
            <rd>{{ rd_2001 }}</rd>
            {%- endif %}
            <name>{{ vrf_def['name'] }}</name>
            {% if vrf_def['address-family'] -%}
            <address-family>
              <ipv4>
                {% if vrf_def['address-family']['ipv4']['route-target'] -%}
                <route-target>
                  {% for export in
vrf_def['address-family']['ipv4']['route-target']['export'] -%}
                  <export>
                    <asn-ip>{{ export['asn-ip'] }}</asn-ip>
                  </export>
                  {%- endfor %}

```

```

                                {% for import in
vrf_def['address-family']['ipv4']['route-target']['import'] -%}
                                <import>
                                <asn-ip>{{ import['asn-ip'] }}</asn-ip>
                                </import>
                                {%- endfor %}
                                </route-target>
                                {%- endif %}
                                {% if vrf_def['address-family']['ipv4']['import'] -%}
                                <import>
                                <map>{{
vrf_def['address-family']['ipv4']['import']['map'] }}</map>
                                </import>
                                {%- endif %}
                                {% if vrf_def['address-family']['ipv4']['mdt'] -%}
                                <mdt>
                                {% if
vrf_def['address-family']['ipv4']['mdt']['default'] -%}
                                <default>
                                <address>{{
vrf_def['address-family']['ipv4']['mdt']['default']['address'] }}</address>
                                </default>
                                {%- endif %}
                                {% if vrf_def['address-family']['ipv4']['mdt']['data']
-%}
                                <data>
                                {% for multicast in
vrf_def['address-family']['ipv4']['mdt']['data']['multicast'] -%}
                                <multicast>
                                <address>{{ multicast['address']
}}</address>
                                <wildcard>{{ multicast['wildcard']
}}</wildcard>
                                </multicast>
                                {%- endfor %}
                                </data>
                                {%- endif %}
                                </mdt>
                                {%- endif %}

```

```

        </ipv4>
      </address-family>
    }%- endif %}
  </definition>
}%- endif %}
}%- endfor %}
</vrf>
</native>
</device>
}%- endfor %}
}%- endfor %}
</devices>
</config>

```

Defining Template Payload

The screenshot shows the ATOM configuration interface for editing a VRF policy. The 'Conditions and Actions' tab is selected, showing a list of conditions on the left and configuration details on the right. The 'Template Payload' field is populated with XML code for a VRF configuration.

Condition Name	Sequence Number
<input checked="" type="checkbox"/> LON2001	1
<input type="checkbox"/> LON2007	1
<input type="checkbox"/> LON2021	1
<input type="checkbox"/> LON2032	1
<input type="checkbox"/> LON2040	1

Condition Details

Condition Name: LON2001

Sequence Number: 1

Scope Details

Condition scope details: Inventory_Data

Condition Match Criteria

Inventory Operator: MATCHES_THE_TEMPLATE_PAYLOAD

Template Payload

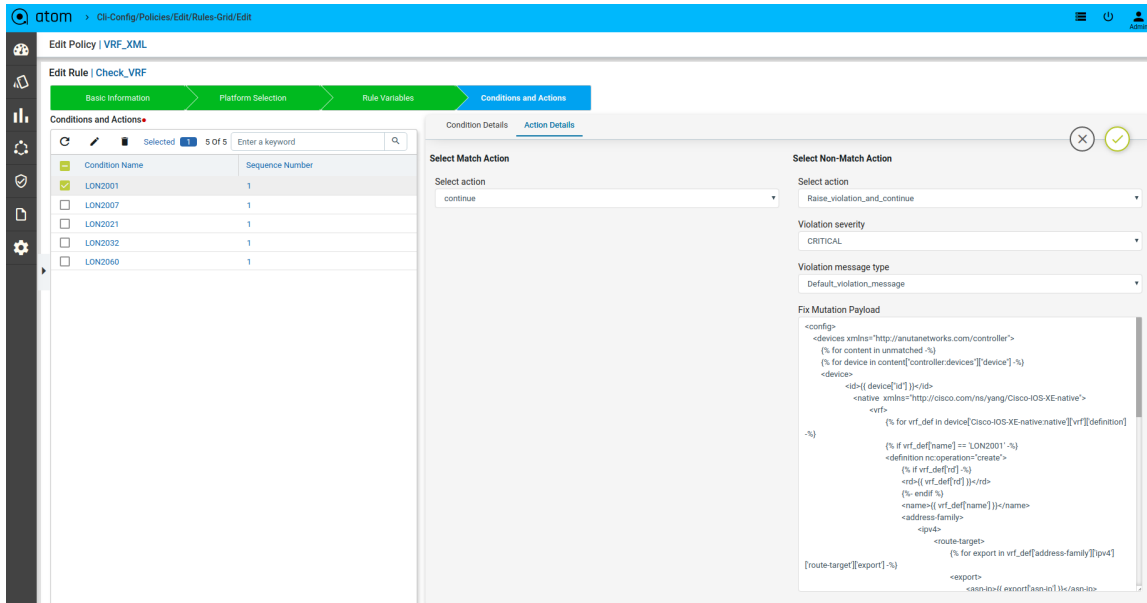
```

<vrf>
  <default>
    <address>239.232.0.1</address>
  </default>
  <data>
    <multicast>
      <address>239.232.1.0</address>
      <wildcard>0.0.0.255</wildcard>
    </multicast>
  </data>
  </vrf>
  <route-target>
    <export>
      <asn-ip>2.2001</asn-ip>
    </export>
    <import>
      <asn-ip>2.2001</asn-ip>
    </import>
  </route-target>
</vrf>
</address-family>
<name>LON2001</name>
<id>2.2001</id>
</definition>
</vrf>
</native>

```

Launch Test Config

Defining Fix Payload



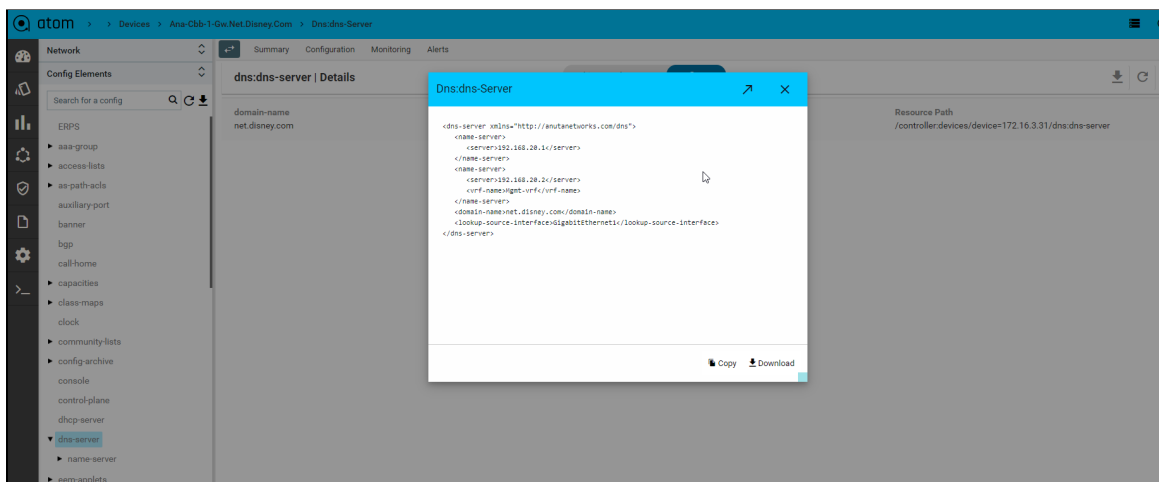
How to derive the XML Template payload

- Navigate to the Device profile page and export the XML template details for the yang parsed entities

Resource Manager → Devices → Select a Device → Configuration → Config Data → Entities → Select Abstract entity → Use Download button to export/copy the XML payload

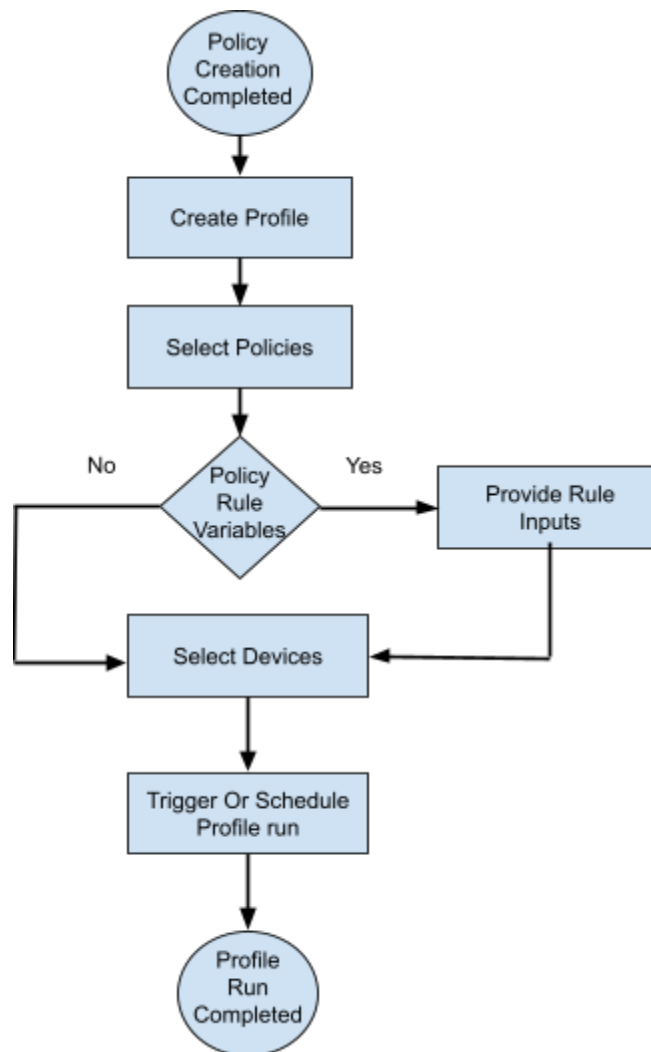
Example : Let's derive domain-name XML Template payload

Navigate to **Devices** → **select a device** → **configuration** -> **Config Data** → **Entities** → **dns-server** → **Export XML** payload using download button.



Profiles

A profile allows one or more Policies to be grouped and executed on one or more devices either on-demand or as per Schedule. Profile execution results in a per-device compliance report included in the execution.



Steps:

- a. Navigate to Resource Manager > Config Compliance -> Profiles
 - Select “+” to Create a Profile
 - ATOM opens up a new wizard and displays 2 sections.
 - Policies - Select one/more policies
 - Devices & Schedule - Select one/more devices or Device groups

b. Create profile by providing name, description and select policy which was created previously IP_Domain_Name.

Select policies

Profile name: Day0_Config

Description: Description of the profile

Select policies

Selected 1 33 Of 33

Name	Description
IP_Domain_Name	Check whether the domain name is present in the

c. Navigate to the next tab, Select devices and schedule. We can select either device(s) from *Devices* or *Device Groups* tab

Select devices and schedule

Devices Device Groups

Selected 1 13 Of 13

Name	Description
AllDevices	All Device Group
Firewall	Firewall Devices Group
Host	Host Devices Group

Edit Profile

Select devices and schedule

Devices Device Groups

Selected 1 22 Of 22

ID	Status	Name	Device Type	Ver
172.16.3.170		172.16.3.170	Cisco CSR 1000V	Cis

d. After device(s) are selected, choose if the compliance checks need to be run against an archived config or current running-configuration of the device. By default Latest From Config Archive is selected.

Schedule: The profile job can be scheduled in Hours or Minutes. Alternatively, a job can be started right away by enabling *Start now* option.

Select Configuration

☐ Latest From Config Archive

☒ Current Config

Schedule

Frequency

0



Hours

0

Minutes

☐ Start now

Or the profile job can triggered at a later point of time using run job icon on the profiles view



Selected 1

Name	Description	Policies	
<input checked="" type="checkbox"/> ip_domain_profile		IP_Domain_Name	

Report

Navigate to Resource Manager > Config Compliance -> reports

Compliance report is generated upon completion of Profile run. For each device, the report lists the compliant and non-compliant policies, rules and conditions .

After profile job is run, audit details can be viewed in Report View

Host Name	Device Type	Severity	Device Compliance Status	Execution	Condition Status	Device Id	Vendor	Policy Name	Rule Name	Condition Name
wnacrp-dtss-0-gw.net.disney.com	Cisco CSR 1000V	Critical	Non Compliant	Successful	Non Compliant	172.16.3.30	Cisco Systems	IP_Domain_N...	Check_IP_D...	Verify_IP_Domain...

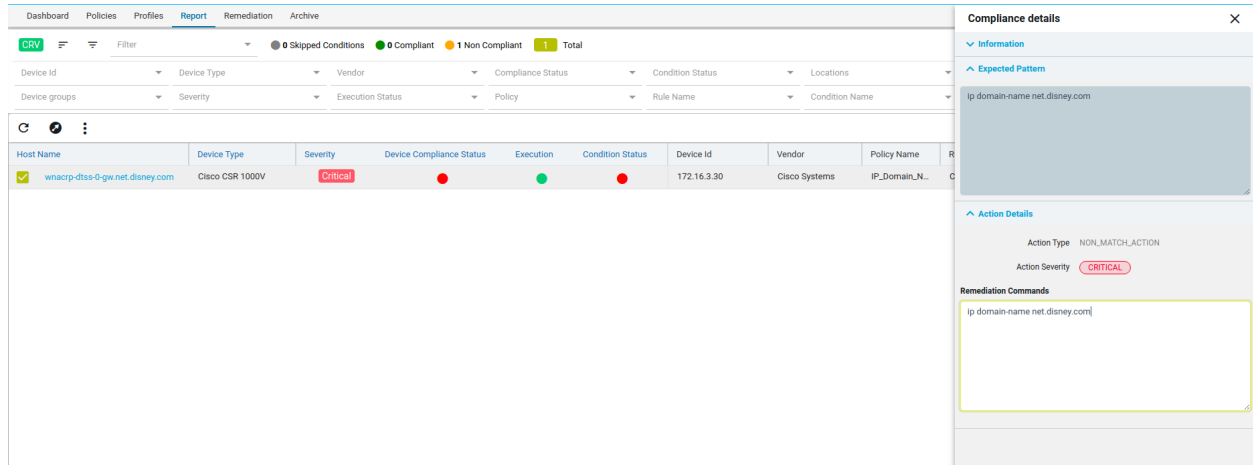
Since *IP_Domain_Name* policy has a condition named *Verify_IP_Domain_name*, where it didn't meet the required criteria. The condition is marked as *Non_compliant*.

Severity: Severity the condition where the condition Match Action or Non Match Action is of type *Raise_violation* or *Raise_violation_and_continue*.

Upon checking the row you can see the expected and the fix commands for that condition along with action-severity, action-type and other metadata related to device & condition.

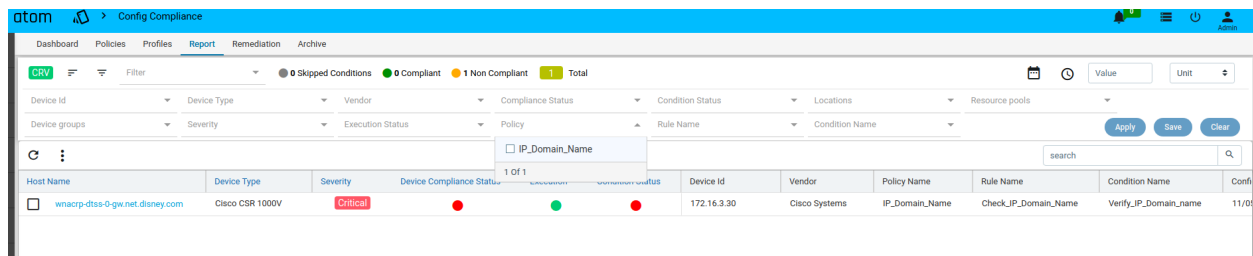
Host Name	Device Type	Severity	Device Compliance Status	Execution	Condition Status	Device Id	Vendor	Policy Name	Rule Name	Condition Name
wnacrp-dtss-0-gw.net.disney.com	Cisco CSR 1000V	Critical	Non Compliant	Successful	Non Compliant	172.16.3.30	Cisco Systems	IP_Domain_N...	Check_IP_D...	Verify_IP_Domain...

Compliance details	
Device ID	172.16.3.30
Device host name	wnacrp-dtss-0-gw.net.disney.com
Device Type	Cisco CSR 1000V
Vendor	Cisco Systems
Device Compliance Status	Non Compliant
Execution Status	SUCCESSFUL
Config Time	Nov 5, 2020, 10:46:16 PM
Policy Name	IP_Domain_Name
Rule Name	Check_IP_Domain_Name
Condition ID	Verify_IP_Domain_name
Condition Status	Non Compliant



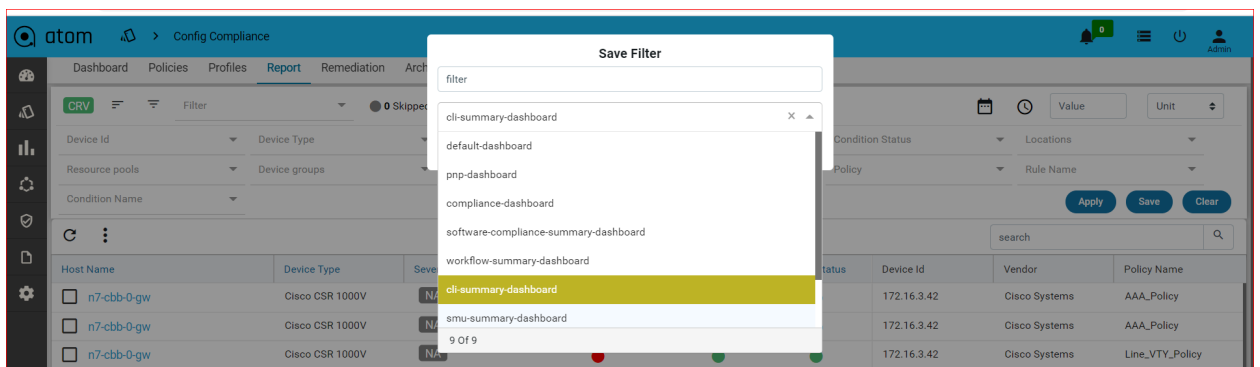
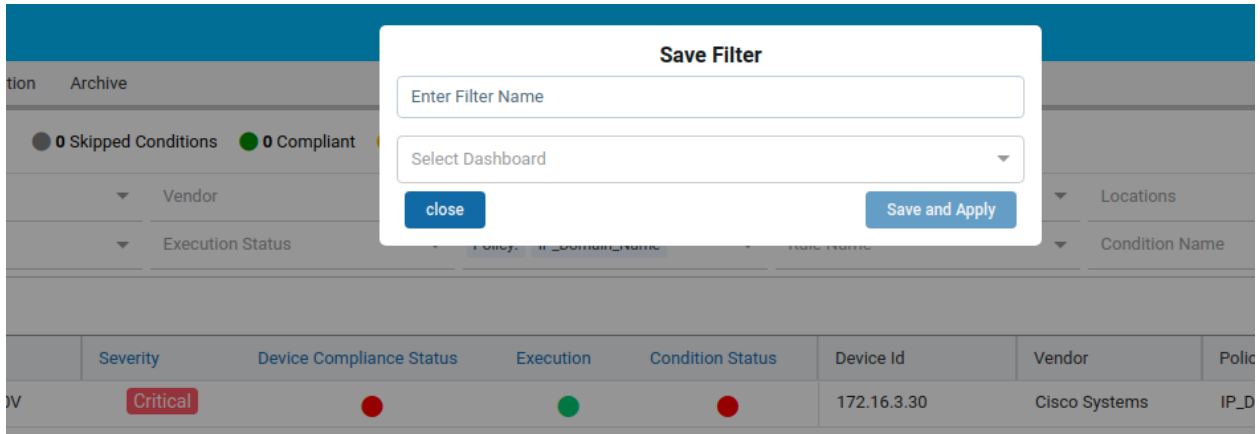
The reports section also facilitates users to filter the results of what user is interested in. The dropdown will display all the possible values for the filters. Users can try out any combination and see the results. By clicking on the **apply** button.

Inorder to revert the filter that are applied you can click on the **clear** button.

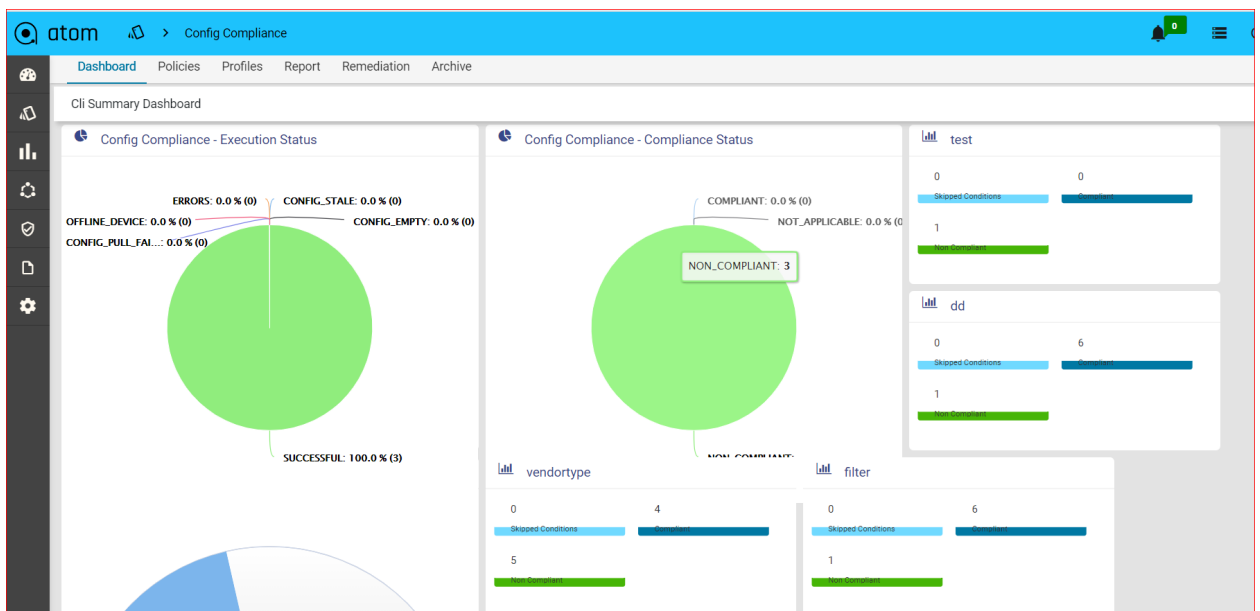


Tired of filtering the results every time for more frequent data. We got you covered ATOM provides an option to save the filter that you applied again with a single click.

- Select filters that you are interested in and click on the **save** button
- This will show a new pop-up box prompting for the filter name and the dashboard where the user wants to pin it.
- Click on the **save and apply** button will save this filter and the resulting data will be populated.

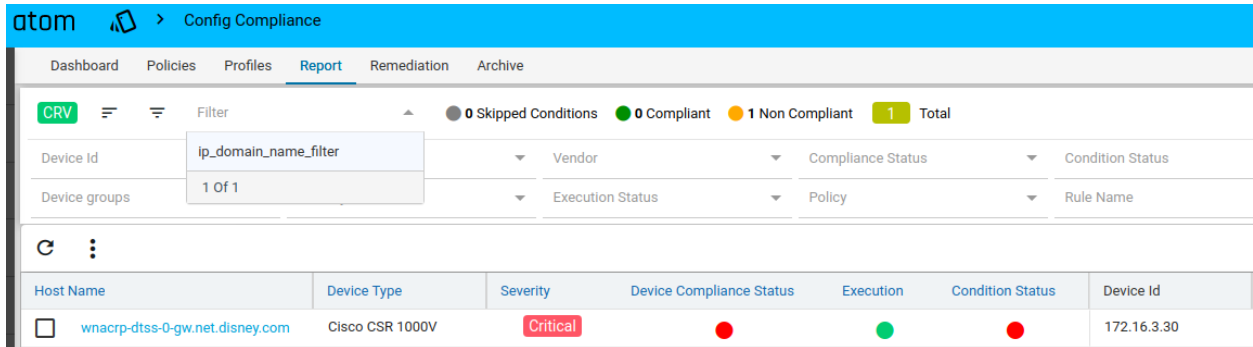


We can pin to the dashboard, upon saving the filter using dropdown. we are able to see the filter under the dashboard.



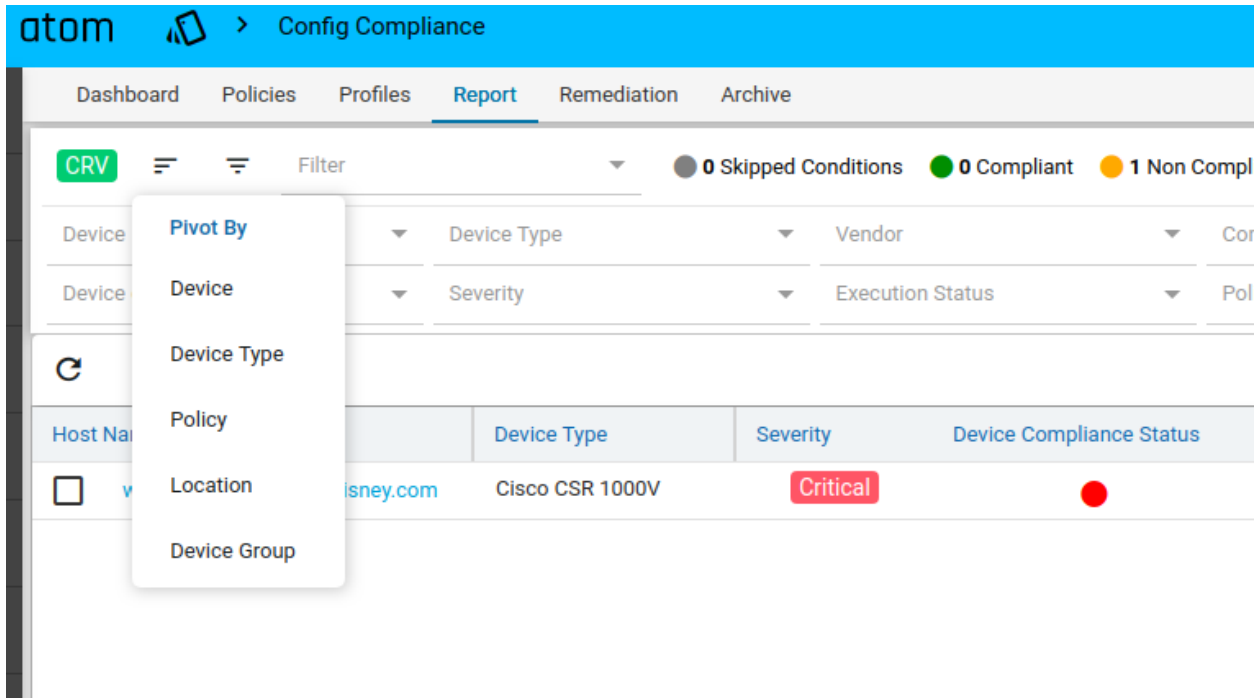
From where I can access these saved filters?

- They are easily accessible. All the filters that are saved are listed under the dropdown on the top.
- Click on the interested filter and you see the data getting filtered.



The screenshot shows the ATOM Config Compliance Report page. The top navigation bar includes Dashboard, Policies, Profiles, Report (selected), Remediation, and Archive. Below the navigation bar, there's a summary section with a green 'CRV' status, a filter dropdown, and statistics: 0 Skipped Conditions, 0 Compliant, 1 Non Compliant, and 1 Total. The filter dropdown is open, showing 'ip_domain_name_filter' and '1 Of 1'. Below the filter, there's a table with columns: Host Name, Device Type, Severity, Device Compliance Status, Execution, Condition Status, and Device Id. The table contains one row with the following data: Host Name: wnacrp-dtss-0-gw.net.disney.com, Device Type: Cisco CSR 1000V, Severity: Critical, Device Compliance Status: (red dot), Execution: (green dot), Condition Status: (red dot), Device Id: 172.16.3.30.

ATOM also provides an option to view the statistics based on Pivot by Device, Device Type, Policy, Location and Device Group.



The screenshot shows the ATOM Config Compliance Report page with the Pivot By dropdown menu open. The dropdown menu lists the following options: Pivot By, Device, Device Type, Policy, Location, and Device Group. The background shows the same summary section and table as the previous screenshot, but the table is partially obscured by the dropdown menu.

Users can opt for any view that they are interested in.

Pivot by device

Device Compliance Status	Severity	Execution	Host Name	Device ID	Device Type	Compliant Policies	Non-Compliant Policies	Compliant Conditions	Non-Compliant Conditions	Vendor
<input type="checkbox"/>	Critical	●	wnacrp-dtss-0-gw-net.disney...	172.16.3.30	Cisco CSR 1000V	0	1	0	1	Cisco Systems

Severity: The aggregated severity of that particular device.

Compliant Policies: The number of policies that are compliant against the device.

Non-Compliant Policies: The number of policies that are non-compliant against the device.

Compliant Conditions: The number of Conditions that are complaint against the device

Non-Compliant Conditions: The number of Condition that are non-compliant against the device

Device ID: Displays all Device Ids for which compliance has run. This is the context column for pivot by device.

Execution status: Based on execution on device it is updated as Successful, errors, stale config, empty config, config pull failed, offline device.

Hostname: Hostname of a device is displayed here.

Device compliance status: Based on compliance run, for a device it is updated as Compliant, non-compliant, Not applicable.

Device Type: The device type of a device is displayed (like cisco csr 1000v , cisco 891).

Vendor: vendors of device are displayed here (like cisco, juniper)

On clicking on the Device ID, Device ID filter gets applied and the user will be navigated to CRV.

Dashboard Policies Profiles Report Remediation Archive										
<div> <div>CRV</div> <div>Filter</div> <div> <div>2 Skipped Conditions</div> <div>3 Compliant</div> <div>22 Non Compliant</div> <div>27 Total</div> </div> </div> <div> <div>Device Id: 172.16.22.101</div> <div>Device Type</div> <div>Vendor</div> <div>Compliance Status</div> <div>Condition Status</div> <div>Locations</div> </div> <div> <div>Resource pools</div> <div>Device groups</div> <div>Severity</div> <div>Execution Status</div> <div>Policy</div> <div>Rule Name</div> </div> <div> <div>Condition Name</div> <div>Apply</div> <div>Save</div> <div>Clear</div> </div>										
<div> <div>search</div> <div>Q</div> </div>										
Host Name	Severity	Execution	Device Type	Device Compliance Status	Condition Status	Device Id	Vendor	Policy Name	Rule Name	Condition Name
<input type="checkbox"/> csr101.anutanetworks.com	Critical		Cisco CSR 1000V			172.16.22.101	Cisco Systems	ACL_on_Line...	Check_Line...	Verify_Line_VTY
<input type="checkbox"/> csr101.anutanetworks.com	Critical		Cisco CSR 1000V			172.16.22.101	Cisco Systems	Clock_Synchr...	NTP_TEMPL...	Check_NTP_Server
<input type="checkbox"/> csr101.anutanetworks.com	Critical		Cisco CSR 1000V			172.16.22.101	Cisco Systems	Clock_Synchr...	NTP_TEMPL...	Check_NTP_Asoo...
<input type="checkbox"/> csr101.anutanetworks.com	Critical		Cisco CSR 1000V			172.16.22.101	Cisco Systems	Clock_Synchr...	NTP_TEMPL...	Check_NTP_ACL
<input type="checkbox"/> csr101.anutanetworks.com	Critical		Cisco CSR 1000V			172.16.22.101	Cisco Systems	Clock_Synchr...	CLOCK_TE...	Check_Summer...
<input type="checkbox"/> csr101.anutanetworks.com	Critical		Cisco CSR 1000V			172.16.22.101	Cisco Systems	Enable_Pass...	Check_pass...	verify_Password...
<input type="checkbox"/> csr101.anutanetworks.com	Critical		Cisco CSR 1000V			172.16.22.101	Cisco Systems	IP_Domain_N...	Check_IP_D...	Verify_IP_Domain...
<input type="checkbox"/> csr101.anutanetworks.com	Critical		Cisco CSR 1000V			172.16.22.101	Cisco Systems	IP_Name_Ser...	Check_IP_N...	IP_Name_Server
<input type="checkbox"/> csr101.anutanetworks.com	NA		Cisco CSR 1000V			172.16.22.101	Cisco Systems	NTP_configur...	NTP_config...	Remove_extra_N...
<input type="checkbox"/> csr101.anutanetworks.com	Critical		Cisco CSR 1000V			172.16.22.101	Cisco Systems	NTP_configur...	NTP_config...	NTP_server_check
<input type="checkbox"/> csr101.anutanetworks.com	NA		Cisco CSR 1000V			172.16.22.101	Cisco Systems	Standard_Aud...	Check_Privil...	verify_VTY
<input type="checkbox"/> csr101.anutanetworks.com	NA		Cisco CSR 1000V			172.16.22.101	Cisco Systems	Standard_Aud...	Check_Privil...	verify_Console
<input type="checkbox"/> csr101.anutanetworks.com	Critical		Cisco CSR 1000V			172.16.22.101	Cisco Systems	Standard_Aud...	Check_vty_s...	verify_session_tl...

Pivot By Policy

<div> <div>PV</div> <div>Policy x</div> <div>Filter</div> <div> <div>0 Skipped Conditions</div> <div>0 Compliant</div> <div>1 Non Compliant</div> <div>1 Total</div> </div> </div> <div> <div>Device Id</div> <div>Device Type</div> <div>Vendor</div> <div>Compliance Status</div> <div>Condition Status</div> </div> <div> <div>Device groups</div> <div>Severity</div> <div>Execution Status</div> <div>Policy</div> <div>Rule Name</div> </div>						
<div> <div>Compliance Status</div> <div>Severity</div> <div>Policy Name</div> <div>Compliant Devices</div> <div>Non-Compliant Devices</div> <div>Non-Compliant Conditions</div> </div>						
<input type="checkbox"/> 	Critical	IP_Domain_Name	0	1	1	

Severity: The aggregated severity of that particular policy.

Compliant Devices: The number of devices that are compliant against the policy.

Non-Compliant Devices: The number of devices that are non-compliant against the policy.

Non-Compliant Conditions: The number of conditions that are non-compliant against the policy.

Policy Name: Displays all policies for which compliance is run. This is the context column for pivot by policy.

Compliance status: Based on policy, it is updated as compliant or non-compliant.

On clicking on the policy, the policy filter gets applied and the user will be navigated to CRV.

CRV

Filter

0 Skipped Conditions

0 Compliant

15 Non Compliant

15 Total

Value

Unit

Device Id

Device Type

Vendor

Compliance Status

Condition Status

Locations

Resource pools

Device groups

Severity

Execution Status

Policy: ACL_on_Line_VTY

Rule Name

Condition Name

Apply

Save

Clear

C

:

search

Q

Host Name	Device Type	Severity	Device Compliance Status	Execution	Condition Status	Device Id	Vendor	Policy Name	Rule Name	Condition Name
<input type="checkbox"/> aancbb-ana-0-gw	Cisco 871	Critical	●	●	●	172.16.1.138	Cisco Systems	ACL_on_Line...	Check_Line...	Verify_Line_VTY
<input type="checkbox"/> aancbb-ana-1gw.net.dianey.com	Cisco 891	Critical	●	●	●	172.16.1.139	Cisco Systems	ACL_on_Line...	Check_Line...	Verify_Line_VTY
<input type="checkbox"/> car161.anustacorp.com	Cisco CSR 1000V	Critical	●	●	●	172.16.16.161	Cisco Systems	ACL_on_Line...	Check_Line...	Verify_Line_VTY
<input type="checkbox"/> car162.anustacorp.com	Cisco CSR 1000V	Critical	●	●	●	172.16.16.162	Cisco Systems	ACL_on_Line...	Check_Line...	Verify_Line_VTY
<input type="checkbox"/> car163.anustacorp.com	Cisco CSR 1000V	Critical	●	●	●	172.16.16.163	Cisco Systems	ACL_on_Line...	Check_Line...	Verify_Line_VTY
<input type="checkbox"/> car164.anustacorp.com	Cisco CSR 1000V	Critical	●	●	●	172.16.16.164	Cisco Systems	ACL_on_Line...	Check_Line...	Verify_Line_VTY
<input type="checkbox"/> ana-buf-4-gw	CiscoIOSXRv9000	Critical	●	●	●	172.16.17.133	Cisco Systems	ACL_on_Line...	Check_Line...	Verify_Line_VTY
<input type="checkbox"/> iosxr-6.5.1-134	CiscoIOSXRv9000	Critical	●	●	●	172.16.17.134	Cisco Systems	ACL_on_Line...	Check_Line...	Verify_Line_VTY
<input type="checkbox"/> asr17_135	CiscoIOSXRv9000	Critical	●	●	●	172.16.17.135	Cisco Systems	ACL_on_Line...	Check_Line...	Verify_Line_VTY
<input type="checkbox"/> eor-cbb-2-gw.net.dianey.com	CiscoIOSXRv9000	Critical	●	●	●	172.16.18.176	Cisco Systems	ACL_on_Line...	Check_Line...	Verify_Line_VTY
<input type="checkbox"/> car101.anustanetworks.com	Cisco CSR 1000V	Critical	●	●	●	172.16.22.101	Cisco Systems	ACL_on_Line...	Check_Line...	Verify_Line_VTY
<input type="checkbox"/> car102.anustacorp.com	Cisco CSR 1000V	Critical	●	●	●	172.16.22.102	Cisco Systems	ACL_on_Line...	Check_Line...	Verify_Line_VTY

Pivot By Device Type:

Severity: The aggregated severity of that particular device type

Device type: Displays all device types available in compliance. This is the context column for pivot by device type.

Vendor: Displays the vendors available in compliance .

Compliant Device: The number of devices that are compliant against the device type

Non-compliant device: The number of devices that are non-compliant against the device type.

Compliant policies: The number of policies that are compliant against the device type

Non-Compliant policies: The number of policies that are non-compliant against the device type

Compliant Condition: The number of conditions that are compliant against the device type.

Non-compliant Condition: The number of conditions that are non-compliant against the device type.

atom

Config Compliance

Dashboard

Policies

Profiles

Report

Remediation

Archive

PV

Device Type

Filter

0 Skipped Conditions

10 Compliant

6 Non Compliant

16 Total

Value

Unit

Device Id

Device Type

Vendor

Compliance Status

Condition Status

Locations

Resource pools

Device groups

Severity

Execution Status

Policy

Rule Name

Condition Name

Apply

Save

Clear

search

Severity	Device Type	Vendor	Compliant Devices	Non-Compliant Devices	Compliant Policies	Non-Compliant Policies	Compliant Conditions	Non-Compliant Co
<input type="checkbox"/> Critical	Cisco CSR 1000V	Cisco Systems	0	1	5	1	6	1
<input type="checkbox"/> High	Juniper vMX	Juniper Networks	0	1	4	1	4	1
<input type="checkbox"/> High	MX204	Juniper Networks	0	1	0	4	0	4

Click on device type> Cisco CSR 1000v here

The screenshot shows the ATOM Config Compliance Report page. The 'Device Type' filter is selected, and the summary table displays the following data:

Severity	Device Type	Vendor	Compliant Devices	Non-Compliant Devices	Compliant Policies	Non-Compliant Policies	Compliant Conditions	Non-Compliant Conditions
Critical	Cisco CSR 1000V	Cisco Systems	0	1	5	1	6	1
High	Juniper MX204	Juniper Networks	0	1	4	1	4	1
High	Juniper MX204	Juniper Networks	0	1	0	4	0	4

On clicking on the Device Type, Device Type filter gets applied and the user will be navigated to CRV.

The screenshot shows the ATOM Config Compliance CRV page. The 'Device Type' filter is set to 'Cisco CSR 1000V'. The table displays the following data:

Host Name	Device Type	Severity	Device Compliance Status	Execution	Condition Status	Device Id	Vendor	Policy Name
n7-cbb-0-gw	Cisco CSR 1000V	NA	●	●	●	172.16.3.42	Cisco Systems	AAA_Policy
n7-cbb-0-gw	Cisco CSR 1000V	NA	●	●	●	172.16.3.42	Cisco Systems	AAA_Policy
n7-cbb-0-gw	Cisco CSR 1000V	NA	●	●	●	172.16.3.42	Cisco Systems	Line_VTY_Policy
n7-cbb-0-gw	Cisco CSR 1000V	NA	●	●	●	172.16.3.42	Cisco Systems	Logging_Policy
n7-cbb-0-gw	Cisco CSR 1000V	NA	●	●	●	172.16.3.42	Cisco Systems	NTP_server_cisco_policy
n7-cbb-0-gw	Cisco CSR 1000V	NA	●	●	●	172.16.3.42	Cisco Systems	SNMP_server_cisco_Policy
n7-cbb-0-gw	Cisco CSR 1000V	Critical	●	●	●	172.16.3.42	Cisco Systems	Cisco_domain_name_policy

Pivot By Location:

The screenshot shows the ATOM Config Compliance dashboard. The top navigation bar includes 'Dashboard', 'Policies', 'Profiles', 'Report', 'Remediation', and 'Archive'. The 'Report' tab is active. Below the navigation bar, there's a summary section with filters and a table of compliance data.

Severity	Location Name	Compliant Devices	Non-Compliant Devices	Compliant Policies	Non-Compliant Policies	Compliant Conditions	Non-Compliant Conditions
<input type="checkbox"/> Critical	india_loc	0	1	5	1	6	1

Severity: The aggregated severity of that particular locations

Location Name: Displays all available locations over which compliance is run. This is the context column for pivot by location.

Compliant Devices: The number of devices that are compliant against locations

Non-Compliant Devices: The number of devices that are non-compliant against locations

Compliant Policies: The number of policies that are compliant against locations

Non-compliant Policies: The number of policies that are non-compliant against locations

Compliant Condition: The number of conditions that are compliant against locations

Non-Compliant Condition: The number of conditions that are non-compliant against locations

Click on any of location name

The screenshot shows the ATOM Config Compliance dashboard with a detailed table of compliance data. The top navigation bar is the same as the previous screenshot. The summary section shows 206 Skipped Conditions, 108 Compliant, 635 Non-Compliant, and 949 Total. The table below lists various locations and their compliance status.

Severity	Location Name	Compliant Devices	Non-Compliant Devices	Compliant Policies	Non-Compliant Policies	Compliant Conditions	Non-Compliant Conditions
<input type="checkbox"/> Critical	San Jose California	0	1	1	11	4	21
<input type="checkbox"/> Critical	San Jose California Delhi Andhra Pradesh Mumbai	0	1	2	10	6	20
<input type="checkbox"/> Critical	location_california	0	4	0	7	8	54
<input type="checkbox"/> Critical	Delhi Andhra Pradesh Mumbai	0	2	0	12	6	44
<input type="checkbox"/> NA	San Jose California	0	0	0	0	0	0
<input type="checkbox"/> NA	location_Milpitas	0	0	0	0	0	0
<input type="checkbox"/> NA	location_USA Los Angeles location_Japan	0	0	0	0	0	0
<input type="checkbox"/> NA	location_USA location_Tokyo Las Vegas	0	0	0	0	0	0
<input type="checkbox"/> NA	location_california	0	0	0	0	0	0

on clicking on the location name, the location filter gets applied and the user will be navigated to CRV.

Dashboard

Policies

Profiles

Report

Remediation

Archive

CRV

Filter

206 Skipped Conditions108 Compliant635 Non Compliant949 Total

Value

Unit

Device Id

Device Type

Vendor

Compliance Status

Condition Status

Locations: location_california

Resource pools

Device groups

Severity

Execution Status

Policy

Rule Name

Condition Name

Apply

Save

Clear

C

:

search

Host Name	Device Type	Severity	Device Compliance Status	Execution	Condition Status	Device Id	Vendor	Policy Name	Rule Name	Condition Name	Configuration
<input type="checkbox"/> Cisco2951	Cisco 2951		NOT_APPLICABLE	CONFIG_E...		10.1.1.1					11/05/20, 12...
<input type="checkbox"/> Cisco3845	Cisco 3845		NOT_APPLICABLE	CONFIG_E...		10.1.1.2					11/05/20, 12...
<input type="checkbox"/> cat385024P-2	cat385024P		NOT_APPLICABLE	CONFIG_E...		10.1.1.3					11/05/20, 12...
<input type="checkbox"/> ciscoAirCT5508K9	ciscoAirCT5508K9		NOT_APPLICABLE	CONFIG_E...		10.1.1.4					11/05/20, 12...
<input type="checkbox"/> cat385024P-3	cat385024P		NOT_APPLICABLE	CONFIG_E...		10.1.1.5					11/05/20, 1.0...
<input type="checkbox"/> cat385024P-1	cat385024P		NOT_APPLICABLE	CONFIG_E...		10.1.1.6					11/05/20, 12...
<input type="checkbox"/> Cisco Virtual ASA	Cisco Virtual ASA		NOT_APPLICABLE	CONFIG_E...		10.1.15.21					11/05/20, 12...
<input type="checkbox"/> ciscoASA5510	ciscoASA5510		NOT_APPLICABLE	CONFIG_E...		10.1.15.23					11/05/20, 12...
<input type="checkbox"/> Cisco Nexus 7004	Cisco Nexus 7004		NOT_APPLICABLE	CONFIG_E...		10.1.15.25					11/05/20, 12...
<input type="checkbox"/> Cisco Nexus 5010 Switch	Cisco Nexus 5010 S...		NOT_APPLICABLE	CONFIG_E...		10.1.15.27					11/05/20, 12...
<input type="checkbox"/> Cisco Nexus 3064 Switch	Cisco Nexus 3064 S...		NOT_APPLICABLE	CONFIG_E...		10.1.15.29					11/05/20, 12...
<input type="checkbox"/> Cisco ASR 9006	Cisco ASR 9006		NOT_APPLICABLE	CONFIG_E...		10.1.15.31					11/05/20, 12...
<input type="checkbox"/> N/A	APIC		NOT_APPLICABLE	CONFIG_E...		10.1.15.34					11/05/20, 12...
<input type="checkbox"/> CiscoSR4331	CiscoSR4331		NOT_APPLICABLE	CONFIG_E...		10.1.2.1					11/05/20, 12...

Adding the same device in multiple resource pools and each RP associated with a different location, all locations will be listed in pivot views.

Adding the same device in two resource pools and associate one RP with location and another RP with no location, only location associated with RP will be listed.

Pivot By Device Group:

atom

Config Compliance

Severity: The aggregated severity of that particular Device group

Device Group: Displays all available device groups over which compliance is run. This is the context column for pivot by device group.

Devices in group: This gives the number of devices in a group

Compliant Devices: The number of devices that are compliant against device groups
Non-Compliant Devices: The number of devices that are non-compliant against device groups
Compliant Policies: The number of policies that are compliant against device groups
Non-compliant Policies: The number of policies that are non-compliant against device groups
Compliant Condition: The number of conditions that are compliant against device groups
Non-Compliant Condition: The number of conditions that are non-compliant against device groups.

Click on any of device groups

Severity	Compliant Devices	Non-Compliant Devices	Device Group	Compliant Policies	Non-Compliant Policies	Compliant Conditions	Non-Compliant Conditions
<input type="checkbox"/> NA	0	3	Layer 2 switch	1	27	9	51
<input type="checkbox"/> NA	0	38	Layer 3 Router	4	78	103	601
<input type="checkbox"/> NA	0	2	Firewall	0	12	5	35
<input type="checkbox"/> NA	0	6	VPN	0	29	17	122
<input type="checkbox"/> NA	0	7	Layer 2/3 switch	3	44	22	96
<input type="checkbox"/> NA	0	1	OfflineDevices	0	11	3	21
<input type="checkbox"/> NA	0	0	grp1	0	0	0	0
<input type="checkbox"/> Critical	0	38	cisco_grp	4	83	105	613
<input type="checkbox"/> Critical	0	1	Host	0	7	2	17
<input type="checkbox"/> NA	0	0	grp1_device	0	0	0	0

On clicking on the device group, the device group filter gets applied and the user will be navigated to CRV.

Host Name	Device Type	Severity	Device Compliance Status	Execution	Condition Status	Device ID	Vendor	Policy Name	Rule Name
<input type="checkbox"/> cat385024P-2	cat385024P		NOT_APPLICABLE	CONFIG_EMPTY		10.1.1.3			
<input type="checkbox"/> cat385024P-3	cat385024P		NOT_APPLICABLE	CONFIG_EMPTY		10.1.1.5			
<input type="checkbox"/> cat385024P-1	cat385024P		NOT_APPLICABLE	CONFIG_EMPTY		10.1.1.6			
<input type="checkbox"/> aancbb-ana-0gw	Cisco 871	NA	●	●	●	172.16.1.138	Cisco Systems	Enable_Password_Encryption	Check_L
<input type="checkbox"/> aancbb-ana-0gw	Cisco 871	Critical	●	●	●	172.16.1.138	Cisco Systems	ACL_on_Line_VTY	Check_L
<input type="checkbox"/> aancbb-ana-0gw	Cisco 871	Critical	●	●	●	172.16.1.138	Cisco Systems	Clock_Synchronization	NTP_TE
<input type="checkbox"/> aancbb-ana-0gw	Cisco 871	Critical	●	●	●	172.16.1.138	Cisco Systems	Clock_Synchronization	NTP_TE
<input type="checkbox"/> aancbb-ana-0gw	Cisco 871	Critical	●	●	●	172.16.1.138	Cisco Systems	Clock_Synchronization	NTP_TE
<input type="checkbox"/> aancbb-ana-0gw	Cisco 871	Critical	●	●	●	172.16.1.138	Cisco Systems	Clock_Synchronization	CLOCK
<input type="checkbox"/> aancbb-ana-0gw	Cisco 871	Critical	●	●	●	172.16.1.138	Cisco Systems	IP_Domain_Name_Global	Check_L
<input type="checkbox"/> aancbb-ana-0gw	Cisco 871	Critical	●	●	●	172.16.1.138	Cisco Systems	IP_Name_Server_Global	Check_L
<input type="checkbox"/> aancbb-ana-0gw	Cisco 871	NA	●	●	●	172.16.1.138	Cisco Systems	NTP_configurations_Global	NTP_co

pivot view :

- Single pivot view can be selected.(no multi pivot view selection)

- Upon selecting a pivot view, further filters like device id, device type, vendor, compliance status, condition status, location, resource pool, device groups, severity, execution status, policy, rule name, condition name can be applied.
- Just with the pivot views, a filter can't be saved, Based on further filters applied, a filter can be saved. The saved filter can be deleted.
- Bulk delete is not supported in pivot views
- Remediation is not supported
- Sorting is not supported on any column in pivot view.
- Searching is not supported in pivot view.
- The counts on top are related to the non- pivot views and labels are from condition status
- Export: based on pivot views, the records can be exported.

CRV(conditional report view) :

Host Name	Device Type	Severity	Device Compliance Status	Execution	Condition Status	Device ID	Vendor	Policy Name	Rule Name	Condition Name
sanbb-ana-1gw.net.daneg.com	Cisco 891	NA	Compliant	Success	Compliant	172.16.1.139	Cisco Systems	ACL_Global	Check_ACL_Configuration	ad_emptpr3_cmb
sanbb-ana-1gw.net.daneg.com	Cisco 891	NA	Compliant	Success	Compliant	172.16.1.139	Cisco Systems	ACL_Global	Check_ACL_Configuration	ad_emptpr3_nervestr
sanbb-ana-1gw.net.daneg.com	Cisco 891	NA	Compliant	Success	Compliant	172.16.1.139	Cisco Systems	ACL_Global	Check_ACL_Configuration	ad_emptpr3_netool
sanbb-ana-1gw.net.daneg.com	Cisco 891	NA	Compliant	Success	Compliant	172.16.1.139	Cisco Systems	ACL_Global	Check_ACL_Configuration	ad_emptpr3_nms
sanbb-ana-1gw.net.daneg.com	Cisco 891	NA	Compliant	Success	Compliant	172.16.1.139	Cisco Systems	ACL_Global	Check_ACL_Configuration	ad_emptpr3_voice
sanbb-ana-1gw.net.daneg.com	Cisco 891	NA	Compliant	Success	Compliant	172.16.1.139	Cisco Systems	ACL_Global	Check_ACL_Configuration	ad_emptpr3_cisco
sanbb-ana-1gw.net.daneg.com	Cisco 891	Critical	Non-Compliant	Success	Non-Compliant	172.16.1.139	Cisco Systems	ACL_Global	Check_ACL_Configuration	ad_emptpr3_daneg
ana-buf-1-gw.net.daneg.com	Cisco CSR 1000V	NA	Compliant	Success	Compliant	172.16.3.45	Cisco Systems	ACL_Global	Check_ACL_Configuration	ad_emptpr3_cmb
ana-buf-1-gw.net.daneg.com	Cisco CSR 1000V	NA	Compliant	Success	Compliant	172.16.3.45	Cisco Systems	ACL_Global	Check_ACL_Configuration	ad_emptpr3_nervestr
ana-buf-1-gw.net.daneg.com	Cisco CSR 1000V	NA	Compliant	Success	Compliant	172.16.3.45	Cisco Systems	ACL_Global	Check_ACL_Configuration	ad_emptpr3_netool
ana-buf-1-gw.net.daneg.com	Cisco CSR 1000V	NA	Compliant	Success	Compliant	172.16.3.45	Cisco Systems	ACL_Global	Check_ACL_Configuration	ad_emptpr3_nms
ana-buf-1-gw.net.daneg.com	Cisco CSR 1000V	NA	Compliant	Success	Compliant	172.16.3.45	Cisco Systems	ACL_Global	Check_ACL_Configuration	ad_emptpr3_voice
ana-buf-1-gw.net.daneg.com	Cisco CSR 1000V	NA	Compliant	Success	Compliant	172.16.3.45	Cisco Systems	ACL_Global	Check_ACL_Configuration	ad_emptpr3_cisco
ana-buf-1-gw.net.daneg.com	Cisco CSR 1000V	Critical	Non-Compliant	Success	Non-Compliant	172.16.3.45	Cisco Systems	ACL_Global	Check_ACL_Configuration	ad_emptpr3_daneg

- The available columns are: Hostname, Severity, Execution, Device Type, Device compliance status, condition status, Device id, Vendor, Policy name, Rule name, Condition name, Configuration retrieved at, Expected pattern, Enforcement time.
- The data can be filtered by applying filters on device id, device type, vendor, compliance status, condition-status, location, resource pool, device groups, severity, execution status, policy, rule name and condition name.
- Time based filtering : For timing based, user can use value and units(Days, weeks, months, hours, minutes) fields.

Date wise filter: User can choose the date from the calendar symbol and click on 'apply' in the calendar.

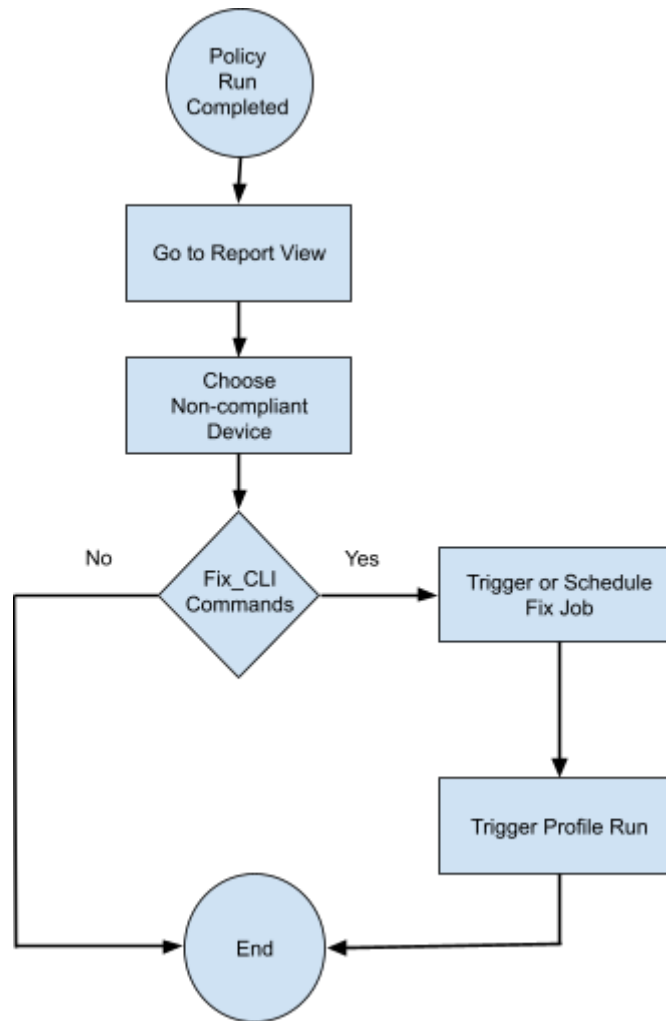
- Count band - Represents counts for skipped, compliant, and non compliant conditions
 - Skipped conditions - Platform mismatch and Execution Status (Stale Config, Empty Config, Erros, Config Pull Failed, Offline Device) fall into this category.
 - Complaint conditions - Based on conditions meeting the criteria
 - Non- compliant conditions -Based on conditions meeting the criteria and violations chosen

- Multi-selection on filters is supported - More than one entry for a given filter can be selected.
- Sorting is enabled on all columns.
- Searching is enabled for all columns.
- The record details are listed when the checkbox for a given entry is selected. Device ID, Device host name, Device type, Vendor, Device compliance status, Execution status, Config time, Policy name, Rule name, Condition ID, Condition status, Expected pattern, Action Details, Remediation commands(if it's non-compliant) are shown as part of details.

Remediation :

Navigate to Resource Manager > Config Compliance -> Remediation

Fix CLI Action will generate the remediation CLI to be applied for each non-compliant device. Users can schedule remediation on one or more devices or execute it right away. For each device selected, ATOM will push remediation CLI to the device.



- a. Select a Report and click on the highlighted arrow for navigating to the Remediation screen.
Only if it is non-compliant, user will be taken to the remediation screen.

CRV

Filter

0 Skipped Conditions

0 Compliant

1 Non Compliant

1 Total

Device Id

Device Type

Vendor

Compliance Status

Condition Status

Device groups

Severity

Execution Status

Policy

Rule Name

Host Name	Severity	Device Type	Device Compliance Status	Condition Status	Execution	Device Id
<div><div></div><div>wncrp-dtss-0-gw.net.disney.com</div></div>	<div>Critical</div>	Cisco CSR 1000V	<div></div>	<div></div>	<div></div>	172.16.3.30

- b. Fix Violations by providing a Job name and verify rule-input values and fix CLI commands under *Fix Configurations*. Click on the tick button to complete fix-job.

Fix Violations

Job name •

fix_ip_domain_name

Devices •

▶ 172.16.3.46

No Rule Inputs

Fix Configurations

ip domain-name anutacorp.com

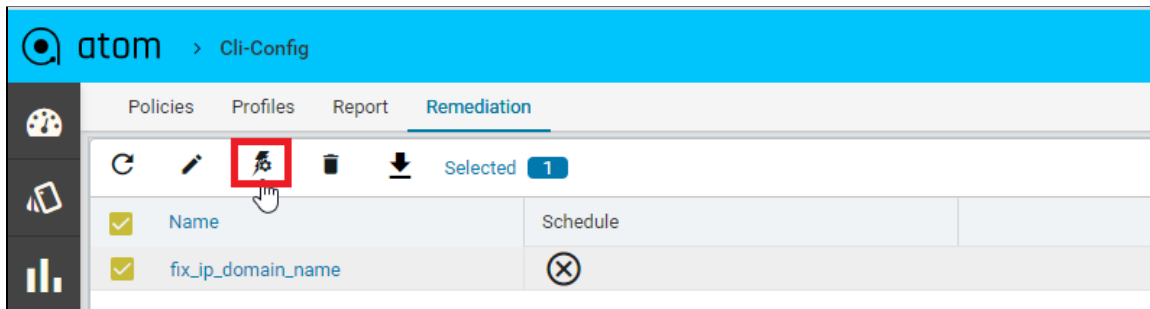
c. Fix Job can be scheduled using Schedule option or can be initiated immediately by enabling Start now. And click on the Tick button to initiate fix-job.

Schedule

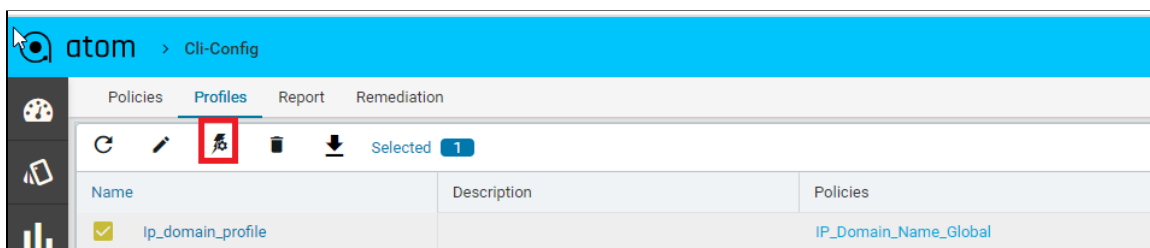
0 0 ☒ Start now

Hours Minutes

1. Fix job can be initiated from the Remediation tab as well at a later point of time.



d. After remediation, trigger profile job and validate the report to see if the device came back as a complaint.



e. Below we can see the device is now back to complaint after fixing violations.

Device Id	Device Type	Vendor	Compliance Status	Condition Status	Locations
wnacrp-dtss-0-gw.net.disney.com	Cisco CSR 1000V		Compliant		

Remediation is not supported when the user is in pivot view.

Bulk delete:

Navigate to Resource Manager > Config Compliance -> reports(CRV)

Host	Device Type	Severity	Device Compliance Status	Execution	Condition Status	Device Id	Vendor	Policy Name
disney.com	Cisco 891	NA	Non Compliant	Compliant	Compliant	172.16.1.139	Cisco Systems	ACL_Global
disney.com	Cisco 891	NA	Non Compliant	Compliant	Compliant	172.16.1.139	Cisco Systems	ACL_Global
aancbb-ana-1gw.net.disney.com	Cisco 891	NA	Non Compliant	Compliant	Compliant	172.16.1.139	Cisco Systems	ACL_Global
aancbb-ana-1gw.net.disney.com	Cisco 891	NA	Non Compliant	Compliant	Compliant	172.16.1.139	Cisco Systems	ACL_Global
aancbb-ana-1gw.net.disney.com	Cisco 891	NA	Non Compliant	Compliant	Compliant	172.16.1.139	Cisco Systems	ACL_Global
aancbb-ana-1gw.net.disney.com	Cisco 891	NA	Non Compliant	Compliant	Compliant	172.16.1.139	Cisco Systems	ACL_Global
aancbb-ana-1gw.net.disney.com	Cisco 891	NA	Non Compliant	Compliant	Compliant	172.16.1.139	Cisco Systems	ACL_Global
aancbb-ana-1gw.net.disney.com	Cisco 891	NA	Non Compliant	Compliant	Compliant	172.16.1.139	Cisco Systems	ACL_Global

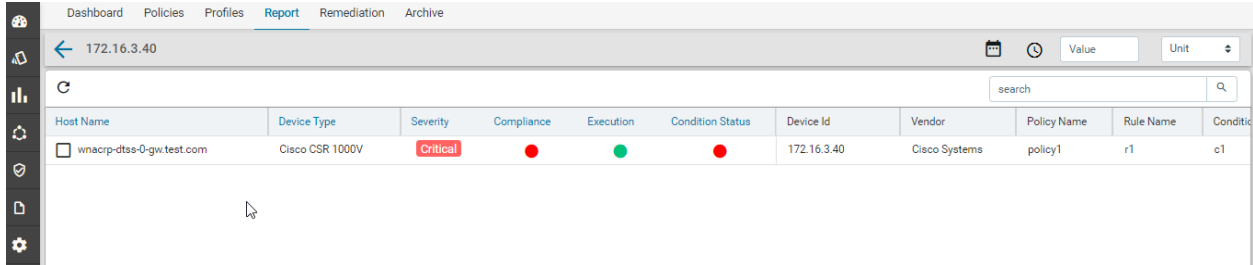
In order to delete the records at shot bulk delete is used.

Based on filters applied, records can be deleted

- If the filter is on Device(device id/ device group/ device Type) And invoke Bulk delete, all the records related to devices are deleted.
- If the filter is on Policy and Bulk delete is invoked,all the records Related to policies across all devices are deleted.(columns like Policy, rule, condition, expected pattern, enforcement time- empty)
- If the filter is on device + policy and bulk delete is invoked, all the records related to policies across all devices are deleted but Device is not deleted.(columns like Policy, rule, condition, expected pattern, enforcement time- empty)
- If the filter is policy+rule+condition, where there are many rules or many conditions, records are deleted at policy level only.(records having the selected policies are deleted irrespective of rule or condition)

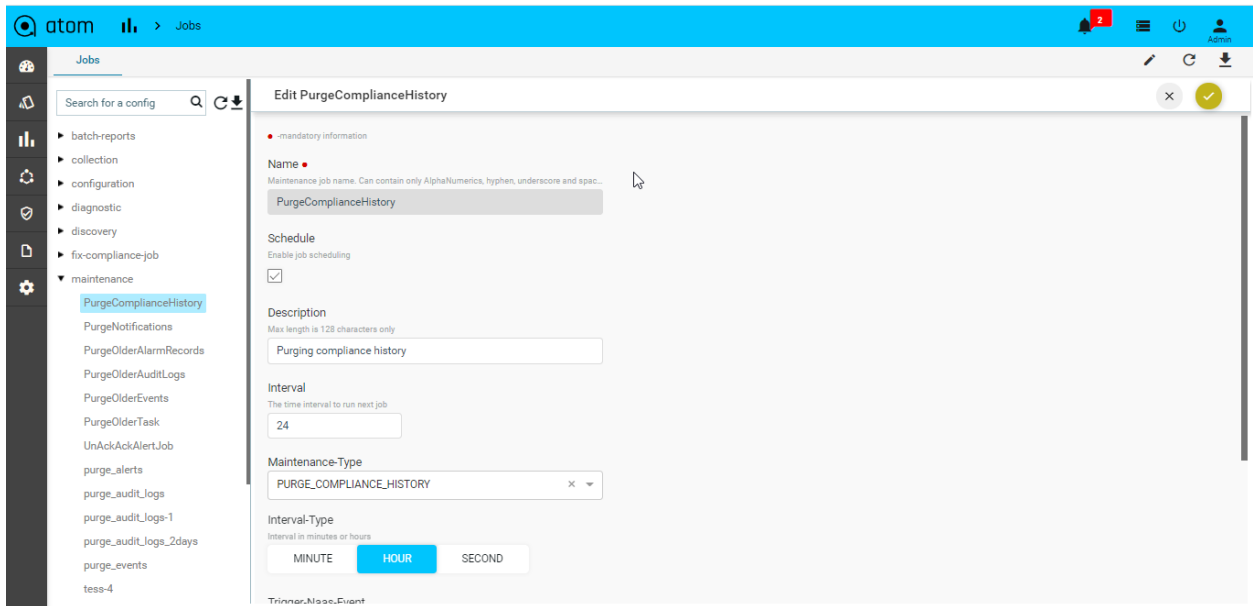
Purge compliance history:

In order to delete history details, we can use purge compliance history under Monitoring--> jobs→ Maintenance→ purge compliance history..



The screenshot shows the ATOM dashboard with the 'Report' tab selected. A table displays compliance data for a specific host. The table has columns for Host Name, Device Type, Severity, Compliance, Execution, Condition Status, Device Id, Vendor, Policy Name, Rule Name, and Condition.

Host Name	Device Type	Severity	Compliance	Execution	Condition Status	Device Id	Vendor	Policy Name	Rule Name	Condition
wnacrp-dtss-0-gw.test.com	Cisco CSR 1000V	Critical	●	●	●	172.16.3.40	Cisco Systems	policy1	r1	c1



The screenshot shows the 'Edit PurgeComplianceHistory' configuration page in the ATOM interface. The page includes a sidebar with a list of jobs and a main form for configuring the purge job.

Jobs List (Left Sidebar):

- batch-reports
- collection
- configuration
- diagnostic
- discovery
- fix-compliance-job
- maintenance**
 - PurgeComplianceHistory**
 - PurgeNotifications
 - PurgeOlderAlarmRecords
 - PurgeOlderAuditLogs
 - PurgeOlderEvents
 - PurgeOlderTask
 - UnAckAckAlertJob
 - purge_alerts
 - purge_audit_logs
 - purge_audit_logs-1
 - purge_audit_logs_2days
 - purge_events
 - tess-4

Edit PurgeComplianceHistory Form:

- Name:** Mandatory information. Maintenance job name. Can contain only AlphaNumerics, hyphen, underscore and space. Value: `PurgeComplianceHistory`
- Schedule:** Enable job scheduling. ☒
- Description:** Max length is 128 characters only. Value: `Purging compliance history`
- Interval:** The time interval to run next job. Value: `24`
- Maintenance-Type:** Value: `PURGE_COMPLIANCE_HISTORY`
- Interval-Type:** Interval in minutes or hours. Options: `MINUTE`, `HOURL` (selected), `SECOND`

Say, the threshold is 60 and it is set for days. When this job is run, it deletes all the record history details that are older than 60 days.

It can be set in hours too.

Export:

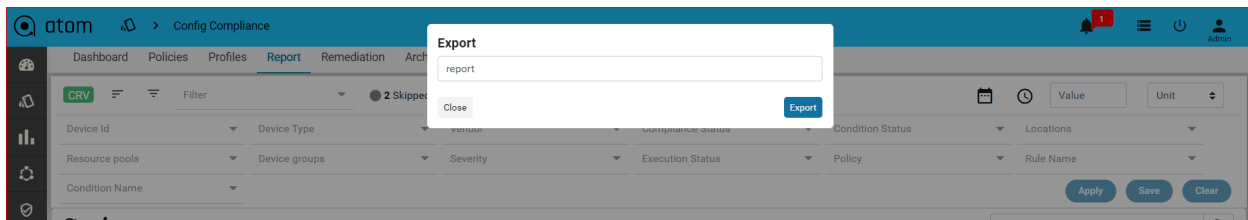
Navigate to Resource Manager > Config Compliance -> reports(CRV/Pivot)

Using export will get report for all the records at one shot

Host	Device Type	Severity	Device Compliance Status	Execution	Condition Status	Device Id	Vendor	Policy Name
disney.com	Cisco 891	NA	Non Compliant	Compliant	Compliant	172.16.1.139	Cisco Systems	ACL_Global
disney.com	Cisco 891	NA	Non Compliant	Compliant	Compliant	172.16.1.139	Cisco Systems	ACL_Global
aancbb-ana-1gw.net.disney.com	Cisco 891	NA	Non Compliant	Compliant	Compliant	172.16.1.139	Cisco Systems	ACL_Global
aancbb-ana-1gw.net.disney.com	Cisco 891	NA	Non Compliant	Compliant	Compliant	172.16.1.139	Cisco Systems	ACL_Global
aancbb-ana-1gw.net.disney.com	Cisco 891	NA	Non Compliant	Compliant	Compliant	172.16.1.139	Cisco Systems	ACL_Global
aancbb-ana-1gw.net.disney.com	Cisco 891	NA	Non Compliant	Compliant	Compliant	172.16.1.139	Cisco Systems	ACL_Global
aancbb-ana-1gw.net.disney.com	Cisco 891	NA	Non Compliant	Compliant	Compliant	172.16.1.139	Cisco Systems	ACL_Global

Upon applying filters and the user does export, then the user can get only those records. Without applying filters, if the user does export, all records are exported.

Provide a name to export file to be saved



Export file will be saved in the archive tab.

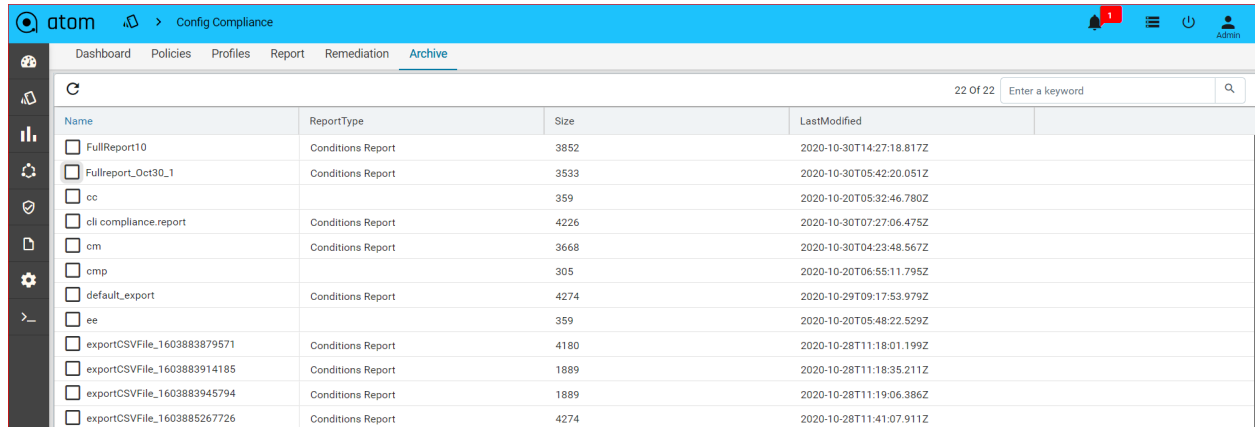
Archive:

Navigate to Resource Manager > Config Compliance -> Archive

From here we can download the report and delete as well. File download format is CSV.

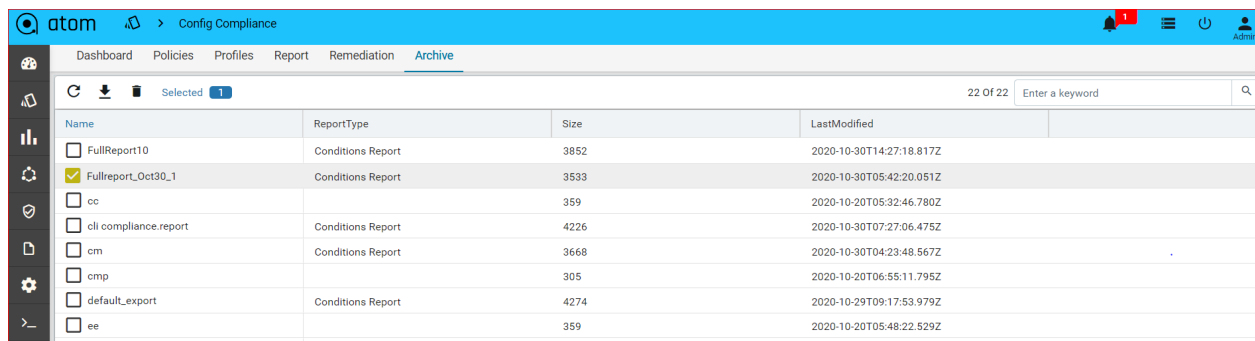
The headers of the downloaded csv report is according to the filter applied.

If pivot views are applied, the headers are according to it.



The screenshot shows the ATOM Config Compliance Archive page. The top navigation bar includes Dashboard, Policies, Profiles, Report, Remediation, and Archive. The Archive tab is selected. The table displays a list of reports with columns for Name, ReportType, Size, and LastModified. The table is sorted by LastModified in descending order.

Name	ReportType	Size	LastModified
<input type="checkbox"/> FullReport10	Conditions Report	3852	2020-10-30T14:27:18.817Z
<input type="checkbox"/> Fullreport_Oct30_1	Conditions Report	3533	2020-10-30T05:42:20.051Z
<input type="checkbox"/> cc		359	2020-10-20T05:32:46.780Z
<input type="checkbox"/> cli compliance.report	Conditions Report	4226	2020-10-30T07:27:06.475Z
<input type="checkbox"/> cm	Conditions Report	3668	2020-10-30T04:23:48.567Z
<input type="checkbox"/> cmp		305	2020-10-20T06:55:11.795Z
<input type="checkbox"/> default_export	Conditions Report	4274	2020-10-29T09:17:53.979Z
<input type="checkbox"/> ee		359	2020-10-20T05:48:22.529Z
<input type="checkbox"/> exportCSVFile_1603883879571	Conditions Report	4180	2020-10-28T11:18:01.199Z
<input type="checkbox"/> exportCSVFile_1603883914185	Conditions Report	1889	2020-10-28T11:18:35.211Z
<input type="checkbox"/> exportCSVFile_1603883945794	Conditions Report	1889	2020-10-28T11:19:06.386Z
<input type="checkbox"/> exportCSVFile_1603885267726	Conditions Report	4274	2020-10-28T11:41:07.911Z



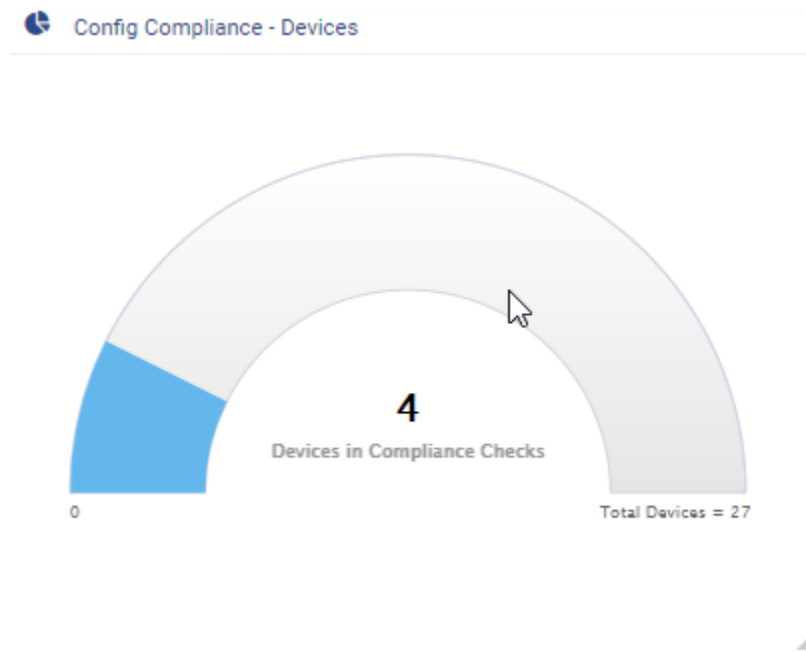
The screenshot shows the ATOM Config Compliance Archive page with the 'Fullreport_Oct30_1' report selected. The table displays the same list of reports as the previous screenshot, but with the selected report highlighted.

Name	ReportType	Size	LastModified
<input type="checkbox"/> FullReport10	Conditions Report	3852	2020-10-30T14:27:18.817Z
<input checked="" type="checkbox"/> Fullreport_Oct30_1	Conditions Report	3533	2020-10-30T05:42:20.051Z
<input type="checkbox"/> cc		359	2020-10-20T05:32:46.780Z
<input type="checkbox"/> cli compliance.report	Conditions Report	4226	2020-10-30T07:27:06.475Z
<input type="checkbox"/> cm	Conditions Report	3668	2020-10-30T04:23:48.567Z
<input type="checkbox"/> cmp		305	2020-10-20T06:55:11.795Z
<input type="checkbox"/> default_export	Conditions Report	4274	2020-10-29T09:17:53.979Z
<input type="checkbox"/> ee		359	2020-10-20T05:48:22.529Z

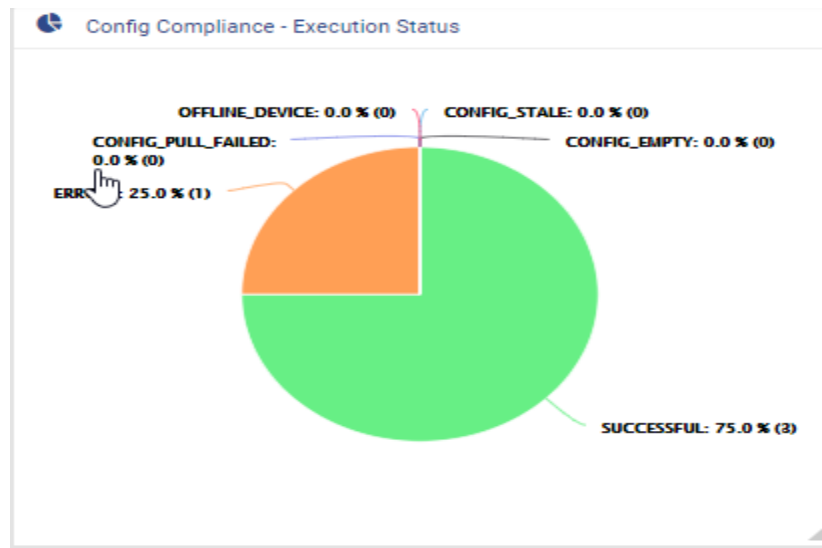
Dashboard

There are 5 Dashboards which gives a quick information about compliance status

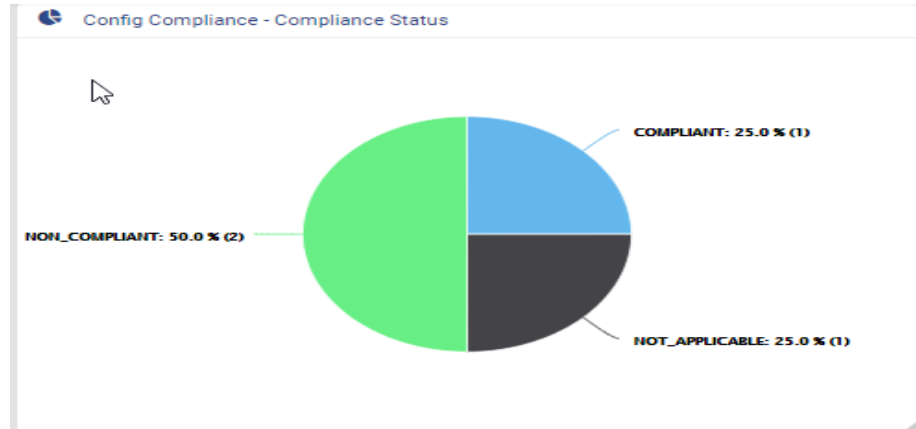
- Config compliance -devices: Representing the number of devices participated in compliance v/s all the available devices in ATOM.



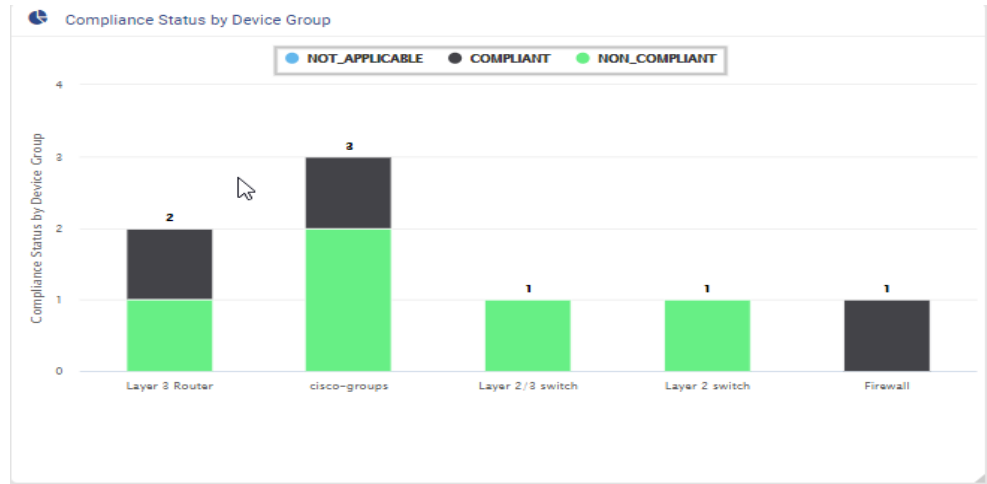
- Config compliance- execution status: A pie chart representing the percentage of execution status in terms of successful, config_empty, config_stale, errors, config_pull_failed, offline_device.



- Config compliance- compliance status: A pie chart representing the percentage of compliance in terms of Complaint, Non-compliant, Not_applicable.

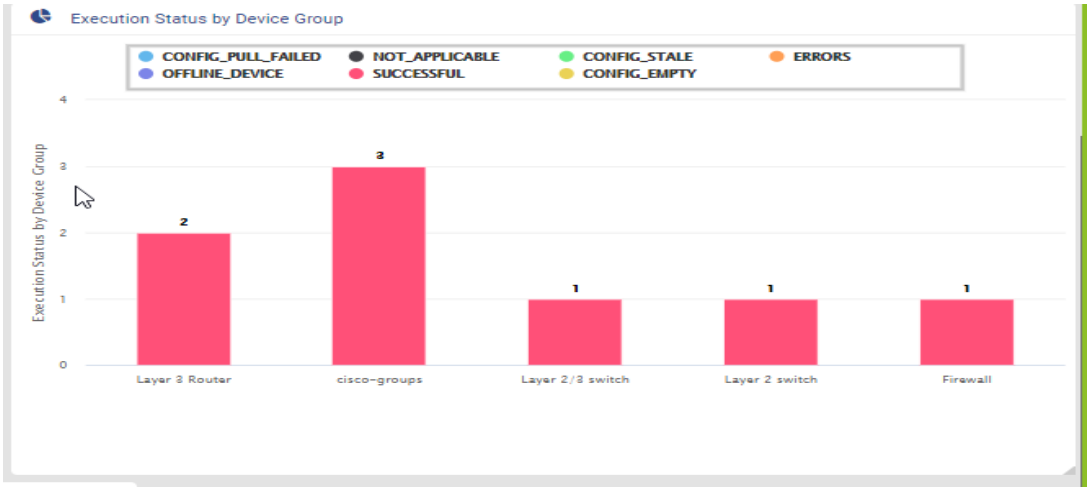


- **Compliance status by group:** A bar graph representing compliance status such as complaint, non-compliant, not_applicable for each and every device group on which compliance profile jobs are run.



- **Execution status by device group:** A bar graph representing execution status such as config pull failed, not applicable, config_stale, errors, offline device, successful, config_empty for each and every device group on which compliance profile jobs are run.

Upon clicking on any of the dashboards, the user is navigated to CRV.



Upon clicking on any of the dashboards, the user is navigated to CRV.

FAQ's

- We have a multi-vendor network, can ATOM help ensure compliance in my environment?

ATOM supports compliance management for Cisco, Juniper and Fortigate at this point in time.

- We want to standardize the network configs even before introducing automation, can ATOM help?

The standard configurations can be defined explicitly in ATOM's compliance framework. ATOM will perform compliance checks against the network and perform remediation in case of non-compliance to standardize the network configurations.

- We already have a platform which checks for compliance, but the remediation is manual, can ATOM help ?

Yes, ATOM is capable of performing remediation on non-compliant devices. The Fix-CLI or derived CLIs can be scheduled or executed immediately to fix all non-compliance issues in the network.

- Can we schedule compliance checks periodically using ATOM ?

Yes, the profiling section in ATOM's compliance framework supports scheduling of compliance checks against a device or a group of devices

- Can ATOM's compliance help in achieving regulatory compliance ?

Yes, based on the policies that regulatory authorities specify, ATOM's compliance management framework can be configured to meet these requirements.

- We have multiple checks that need to be run on the network, can ATOM help handle this scenario ?

ATOM's profiling section supports grouping of multiple policies

Appendix

- Writing Jinja template configurations based on Test Result output

Below is a sample output from the test result obtained in Launching Test Config. This helps writing jinja2 templates as required in use case requirements.

```
{
  "compliance-policies": {
    "highest-severity": "",
    "rule-violation-count": 0,
    "compliance-status": "compliant",
    "compliant-rules-output": {
      "violated-conditions": "",
      "device-compliance-condition-output": {
        "block-start-unmatched-content": "<![CDATA[]]>",
        "block-start-condition-search-output": "<![CDATA[{
          \"block_start_matched_contents\" : []
        }]]>",
        "condition-search-output": "<![CDATA[{
          \"matched_contents\" : [ {
            \"groups\" : [ {
              \"index\" : 1,
              \"grep_content\" : \"1.1.1.1\",
              \"grep_group\" : 1
            } ]
          }, {
            \"groups\" : [ {
              \"index\" : 1,
              \"grep_content\" : \"2.2.2.2\",
              \"grep_group\" : 1
            } ]
          } ]
        }]]>",
        "total-block-count": 2,
        "aggregated-condition-ouput": "<![CDATA[{
          \"condition_contents\" : [ {
            \"condition_id\" : null,
```

```

        "block_start_matched_content" : null,
        "block_start_unmatched_content" : null,
        "unmatched_content" : null,
        "matched_content" : null
    } ]
    ] ]>",
    "template-substituted-content": "<![CDATA[ntp server (?!10.0.0.1)(\d+.\d+.\d+.\d+)]>",
    "block-unmatch-count": 0,
    "cli-match-output": "<![CDATA[ntp server 1.1.1.1
                        ntp server 2.2.2.2
    ] ]>",
    "condition-status": true,
    "unmatched-content": "<![CDATA[]]>",
    "id": "Remove_NTP_Extra_Config",
    "block-match-count": 2,
    "cli-unmatch-output": "<![CDATA[]]>"
},
"name": "test-condition",
"failed-conditions": ""
}
}
}

```

Keys	Condition value	Jinja2 template
matched_contents - when matches with regex in the condition value with the test configuration.	ntp server (?!10.0.0.1)(\d+.\d+.\d+.\d+)	{% for content in matched_contents -%} {% for group in content["groups"] -%} no ntp server {{ group["grep_content"] }} {%- endfor %} {% endfor %}
unmatched_contents - when matches with the regex and does not match with the block config in the condition value with the test configuration.	line vty (.*) session-timeout {{ session_timeout }} exec-timeout {{ exec_timeout }} 0	{% for content in unmatched_contents %} {% for group in content["groups"] %} line vty {{ group["grep_content"] }} session-timeout {{ session_timeout }} exec-timeout {{ exec_timeout }} 0

		<pre>exit {% endfor %} {% endfor %}</pre>
<p>condition_contents - condition1 captured data will be accessible to condition2 using condition_contents.</p>	<p>condition1: interface Loopback0 ip address (\d+.\d+.\d+.\d+) (\d+.\d+.\d+.\d+)</p> <p>condition2: router ospf (.*) router-id {{ condition_contents[0]["matched_content"]["matched_contents"][0]["groups"][0]["grep_content"] }}</p>	<pre>{% for content in unmatched_contents %} {% for group in content["groups"] %} router ospf {{ group["grep_content"] }} router-id {{ condition_contents[0]["matched_content"]["matched_contents"][0]["groups"][0]["grep_content"] }} exit {% endfor %} {% endfor %}</pre>