

# Element Configurations

## Overview

This guide explains how to use FLUX to create task flows that extract device configurations and perform automated audits. You'll learn how to configure audit parameters, set up automated extraction tasks, and review audit results.

## Prerequisites

Before beginning, ensure you have:

- Access to FLUX with appropriate permissions
- Device profiles configured in your system
- Understanding of automation and processing models in FLUX

## Step 1: Configure Element Parameters

### Create a Parameter Group

First, add a parameter group to your element's profile to store configuration audit settings.

Grupos	Colapsado	Repetible	Last change	Interno	¿Eliminar?
Auditoria de Configuraciones Cisco:: org: zequenze	<input type="checkbox"/>	<input type="checkbox"/>	4 de Mayo de 2023 a las 13:01	<input type="checkbox"/>	<input type="checkbox"/>

+ Add

### Define Audit Parameters

Create individual parameters for each configuration aspect you want to evaluate during the audit process.

1 resultado

Parameters									
Nombre	Variable name	Tipo	Pre-determinado	Activo	Sólo-lectura	Requerido	Métrica	¿Eliminar?	
<a href="#">Cantidad de Interfaces</a>	<a href="#">audit.interfaces</a>	Multi-value x ▾	<a href="#">empty</a>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<a href="#">Editar</a>	<input type="checkbox"/>
<a href="#">Dominio valido</a>	<a href="#">audit.valid_domain</a>	Multi-value x ▾	<a href="#">empty</a>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<a href="#">Editar</a>	<input type="checkbox"/>
<a href="#">Interfaz Virtual</a>	<a href="#">audit.virtual_interface</a>	Multi-value x ▾	<a href="#">empty</a>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<a href="#">Editar</a>	<input type="checkbox"/>
<a href="#">Custom Hop</a>	<a href="#">audit.custom_hop</a>	Multi-value x ▾	<a href="#">empty</a>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<a href="#">Editar</a>	<input type="checkbox"/>
<a href="#">Route 0.0.0.0/0</a>	<a href="#">audit.route</a>	Multi-value x ▾	<a href="#">empty</a>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<a href="#">Editar</a>	<input type="checkbox"/>
<a href="#">Hostname 01</a>	<a href="#">audit.valid_hostname</a>	Multi-value x ▾	<a href="#">empty</a>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<a href="#">Editar</a>	<input type="checkbox"/>
<a href="#">Has logging interface</a>	<a href="#">audit.has_logging_interface</a>	Multi-value x ▾	<a href="#">empty</a>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<a href="#">Editar</a>	<input type="checkbox"/>
<a href="#">Has vrf interface</a>	<a href="#">audit.has_vrf_interface</a>	Multi-value x ▾	<a href="#">empty</a>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<a href="#">Editar</a>	<input type="checkbox"/>
<a href="#">Has gigabit interface</a>	<a href="#">audit.has_gigabit_interface</a>	Multi-value x ▾	<a href="#">empty</a>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<a href="#">Editar</a>	<input type="checkbox"/>
<a href="#">Has gigabit two interface</a>	<a href="#">audit.has_gigabit_two_interface</a>	Multi-value x ▾	<a href="#">empty</a>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<a href="#">Editar</a>	<input type="checkbox"/>

## Step 2: Create the Audit Task

Your audit task must perform three key functions:

1. Extract the device configuration
2. Parse the configuration into a manageable format
3. Apply audit conditions using the parameters defined in Step 1

## Extract Device Configuration

Create an automation model that executes a command on the target device to retrieve its configuration output.

**Example:** The following automation model extracts device configuration data.

Nombre:	<input type="text" value="Cisco show running-config"/>		
Short-name:	<input type="text" value="cisco-config"/>		
<input type="checkbox"/> Es público	<small>Public automation Models will be available on sub-organizations too.</small>		
Versión:	<input type="text" value="1.0"/>		
Clase:	<input type="text" value="CLI commands"/>		
Dirección:	<input type="text" value="Out"/>		
Grupos:	<input type="text" value="Click for options"/>		
Is valid:	<span style="color: green;">✔</span> Si - CLI model is valid		
Validation date:	4 de Mayo de 2023 a las 13:09		
Model file:	<input type="text" value="Click for options"/>		
<small>Set a file to load the model data from . Once data is loaded any modification made on the 'model' field below will take precedence over the model data from the selected file.</small>			
Model data:	<table><tr><td>1</td><td>show running-config</td></tr></table>	1	show running-config
1	show running-config		

# Parse the Configuration

Create a processing model to parse the extracted configuration data into a structured format.

**Example:** The following processing model is configured to parse block text files (such as YAML files).

<input type="button" value="↓"/>	<b>Block configuration parser@1.0</b> <small>sys</small>	Clase: <input type="text" value="Block configuration"/>	ID: 61	<input type="button" value="Attach"/>	<input type="button" value="Refrescar"/>		
<small>Creado: 4 de Mayo de 2023 a las 13:03   Last change: 10 de Mayo de 2023 a las 11:02</small>							
Nombre:	<input type="text" value="Block configuration parser"/>						
Short-name:	<input type="text" value="block-config-parser"/>						
<input checked="" type="checkbox"/> Es público	<small>Public automation Models will be available on sub-organizations too.</small>						
Versión:	<input type="text" value="1.0"/>						
Clase:	<input type="text" value="Block configuration"/>						
Dirección:	<input type="text" value="In"/>						
Grupos:	<input type="text" value="Click for options"/>						
Is valid:	<span style="color: green;">✔</span> Si - Configuration block model is valid						
Validation date:	4 de Mayo de 2023 a las 13:03						
Model file:	<input type="text" value="Click for options"/>						
<small>Set a file to load the model data from . Once data is loaded any modification made on the 'model' field below will take precedence over the model data from the selected file.</small>							
Model data:	<table><tr><td>1</td><td></td></tr></table>					1	
1							

For more information about processing models, see the [Models](#) documentation.

# Configure Audit Conditions

In your task processing script, define the configuration data and audit conditions using the element's parameters.

Processing script:

```
1 config.version = '1.7'
2 config.section = 'core'
3 config.required_approval_percentage = 85
4
5 settings['audit.interfaces'] = config.check_key_occurrences("interface.*", ">=", 90, required_for_audit=True)
6 settings['audit.valid_domain'] = config.compare_string_value('domain', 'startswith', 'zequenze', required_for_audit=True)
7 settings['audit.virtual_interface'] = config.complex_compare('interface.x', 'comienza con Virtual', required_for_audit=True)
8 settings['audit.custom_hop'] = config.complex_compare('route.x', 'contiene un nexthop a la ip 172.16.254.1', required_for_audit=True)
9 settings['audit.route'] = config.check_key_matches('route.0.0.0.0/0', required_for_audit=True)
10 settings['audit.valid_hostname'] = config.compare_string_value('hostname', 'contains', '01', required_for_audit=True)
11 settings['audit.has_logging_interface'] = config.check_key_matches("interface.*.logging", required_for_audit=True)
12 settings['audit.has_vrf_interface'] = config.check_key_matches("interface.*.vrf", required_for_audit=True)
13
14 settings['audit.result'] = config.audit_value
```

Please configure the script that will be used to process the results from the automation process and the collected data retrieved from the managed element. Additional documentation of the processing scripts can be found on this [link](#)

For detailed implementation guidance, refer to the [PublicElementConfig](#) documentation.

## Step 3: Review Audit Results

Once the audit task has executed, you can view the results in the Element settings interface.

The screenshot shows the Element settings interface for 'csr01.dev.zequenze.com'. The interface includes a navigation bar with tabs for Settings, Configs, Children, Setup, Logs, Métricas, API logs, Tasks, CLI, and Raw settings. The main content area displays the 'Auditoria de Configuraciones Cisco' section. The results are as follows:

Item	Status	Message
Cantidad de Interfaces:	Fail	4 unique occurrences of pattern 'interface.' were detected
Dominio valido:	OK	Esta prueba evalua la condicion determinada...
Interfaz Virtual:	OK	
Custom Hop:	OK	
Route 0.0.0.0/0:	OK	
Hostname 01:	OK	
Has logging interface:	OK	
Has vrf interface:	OK	
Has gigabit interface:	OK	
Has gigabit two interface:	Fail	Key pattern 'interface.GigabitEthernet2.*' did not match with any key in config
Final Result:	Fail	

## Running Audits on Historical Configurations

You can configure tasks to run audits against previously stored device configurations instead of extracting a new configuration each time.

## Implementation

To audit a historical configuration:

1. In the task's management script, set the `automation_custom_output` variable
2. Populate this variable with the stored configuration data using the `last_config` method from the [PublicElementConfig](#) class

This approach simulates the output of the automation model using previously extracted data.

**Example:** Configuration for auditing historical data.

```
Management script: 1 context['automation_custom_output'] = config.get_last_config()
```

## Related Documentation

- [Models](#) - Comprehensive guide to automation and processing models
- [PublicElementConfig](#) - API reference for configuration management

---

Revision #2

Created 2026-02-13 23:08:22 UTC by ipena@zequenze.com

Updated 2026-03-05 22:30:35 UTC by ipena@zequenze.com