# 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

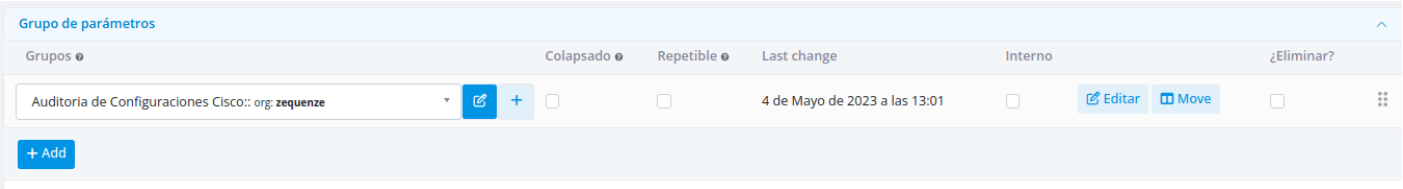
Before beginning, ensure you have:

- 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

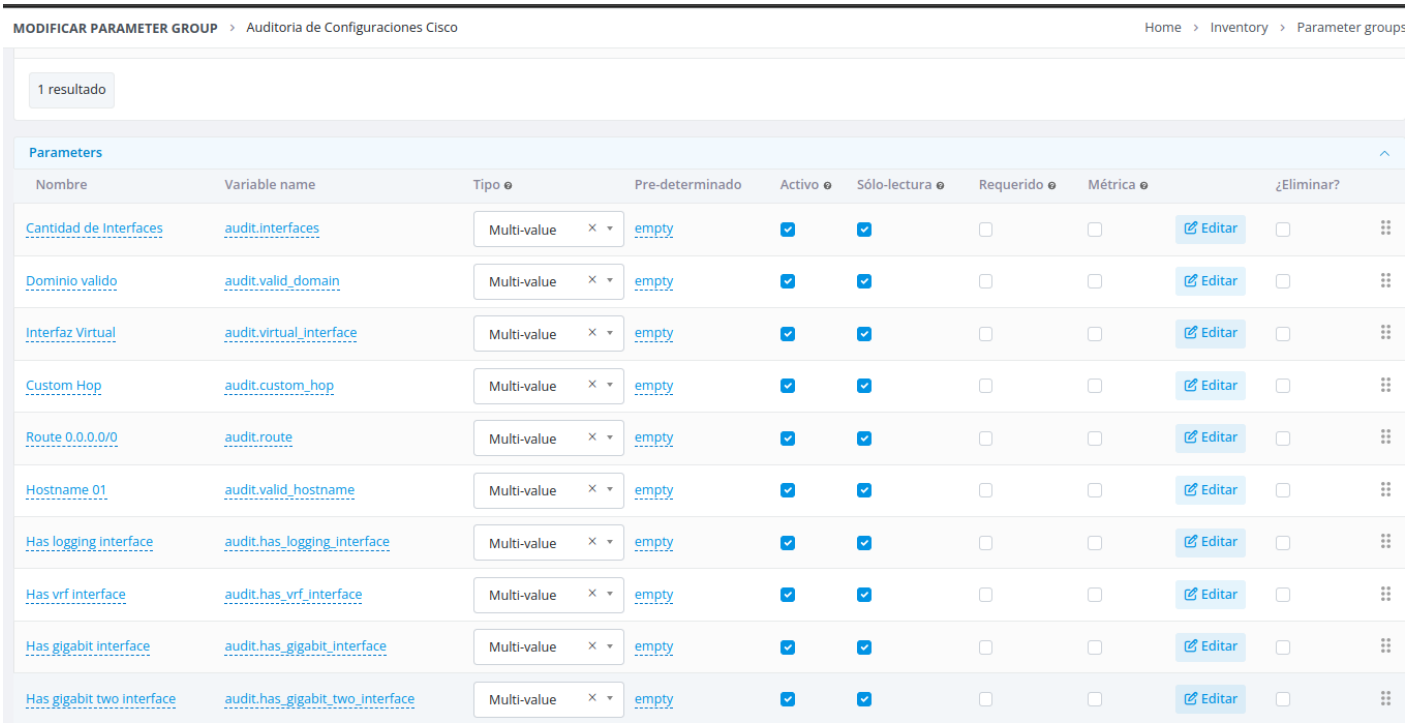
### Create a Parameter Group

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




### Define Audit Parameters

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




# Step 2: Create the Audit Task

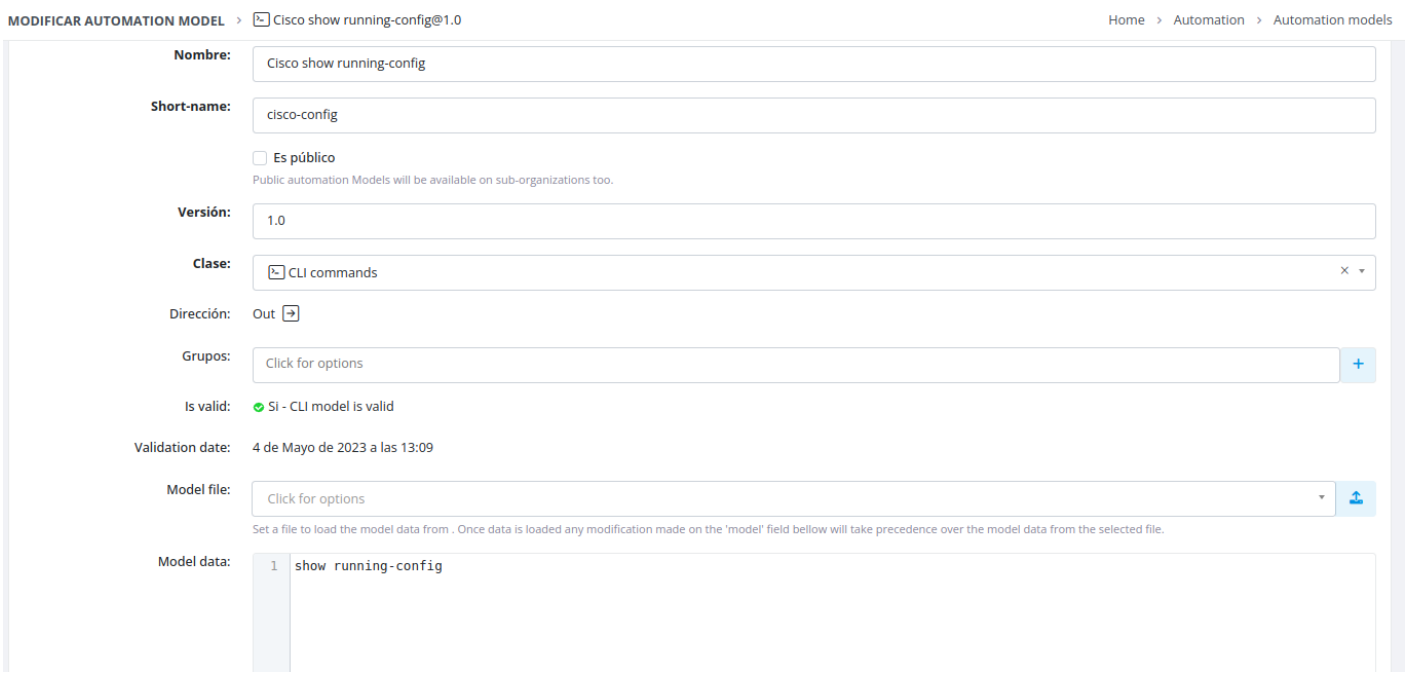
Your audit task must perform three key functions:

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.

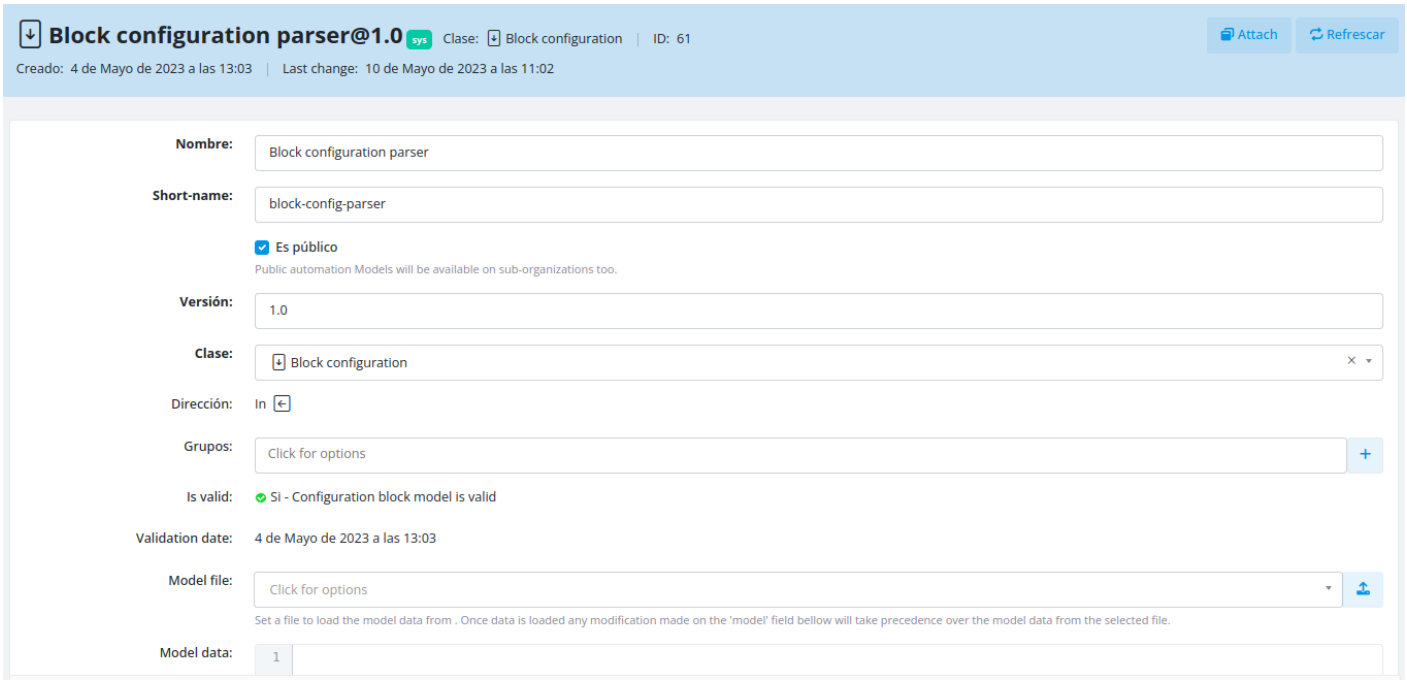
**Example:** The following automation model extracts device configuration data.




# 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).




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:

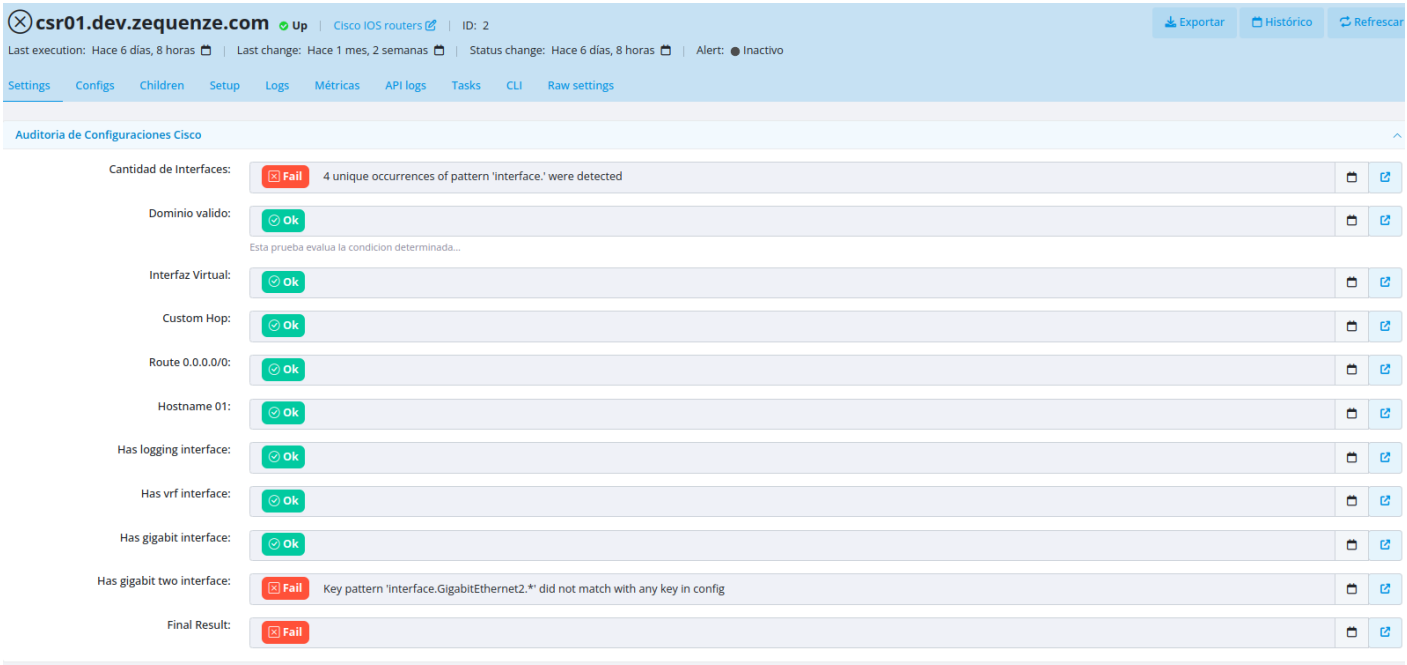
```
config.version = '1.7'
config.section = 'core'
config.required_approvement_percentage = 85

settings['audit.interfaces'] = config.check_key_occurrences("interface.*", ">=", 90, required_for_audit=True)
settings['audit.valid_domain'] = config.compare_string_value('domain', 'startswith', 'zequenze', required_for_audit=True)
settings['audit.virtual_interface'] = config.complex_compare('interface.x', 'comienza con Virtual', required_for_audit=True)
settings['audit.custom_hop'] = config.complex_compare('route.x', 'contiene un nexthop a la ip 172.16.254.1', required_for_audit=True)
settings['audit.route'] = config.check_key_matches('route.0.0.0.0/0', required_for_audit=True)
settings['audit.valid_hostname'] = config.compare_string_value('hostname', 'contains', '01', required_for_audit=True)
settings['audit.has_logging_interface'] = config.check_key_matches("interface.*.logging", required_for_audit=True)
settings['audit.has_vrf_interface'] = config.check_key_matches("interface.*.vrf", required_for_audit=True)

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.




# 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