

Cisco Crosswork AI Configuration Drift Detection Release 2.0 for NSO

Table of Contents

Configuration Drift Detection . . . 1
- What is Configuration Drift Detection? . . . 1
- Configuration Drift Detection capabilities . . . 1

Prerequisites . . . 2
- Download the CWAICTL CLI tool . . . 2

Configuration Drift Detection workflow . . . 3

Set up device and logical groups . . . 4
- Create logical groups using NSO device groups . . . 4
- Manually create a logical group . . . 5

Train the model . . . 7

Run inference to detect configuration anomalies . . . 10
- Access Configuration Drift Detection agent . . . 10
- Analyze configurations on-demand for specific device names . . . 11
- Analyze configurations on-demand for a logical group . . . 14
- Analyze new configurations on-demand . . . 16
- Schedule inference to run later . . . 18
- Manage anomaly feedback . . . 19
  - Download anomaly feedback CSV . . . 19
  - Update the anomaly feedback CSV . . . 19
  - Import anomaly feedback CSV . . . 20
  - Verify anomaly feedback . . . 20
  - Reverse anomaly feedback . . . 20
  - Anomaly feedback behavior and limitations . . . 20

Interpret anomaly detection . . . 21
- Types of anomalies . . . 21
- Interpreting results . . . 21

Agent management with CWAICTL . . . 22
- Logical groups . . . 22
- Add configuration files . . . 23
- Train the model . . . 23
- Analyze configurations by running inference tasks . . . 24
- Delete group . . . 25
- Reset group repository . . . 25

Troubleshooting Configuration Drift Detection agent . . . 26
- Error: "No groups found" . . . 26
- Training job times out . . . 26

Configuration Drift Detection

This document explains how to use Crosswork AI Configuration Drift Detection with Cisco Network Services Orchestrator (NSO). It is intended for network engineers and administrators who use NSO and want to proactively monitor, audit, and remediate configuration drift in their networks.

What is Configuration Drift Detection?

Crosswork AI Configuration Drift Detection uses machine learning to identify unintended changes or anomalies in network device configurations. The Configuration Drift Detection (CDD) agent helps maintain network consistency and compliance by comparing device configurations against learned patterns, rather than relying only on pre-defined templates or manual checks.

Traditional configuration management often depends on manual CLI operations or static templates, which can be error-prone, hard to scale, and challenging in multi-vendor environments due to differences in device syntax and behavior. The CDD agent addresses these challenges by learning typical configuration patterns for each network role and vendor combination. It then detects anomalies by comparing configurations against these learned patterns, enabling early detection and intelligent review of configuration changes.

After anomalies are detected, users can review the results and provide feedback for anomalies that do not require further action. This feedback helps reduce repeated noise in future inference results for the same logical group.

Configuration Drift Detection capabilities

Crosswork AI Configuration Drift Detection Agent enhances network reliability and operational efficiency by providing:

- **Machine learning-based verification**: Complements existing template and change management approaches by automatically learning configuration patterns without requiring manual template creation.
- **Advanced anomaly detection**: Evaluates new or changed configurations against learned patterns to identify deviations. The agent flags configuration lines that differ from what is expected, quickly highlighting potential issues.
- **Anomaly feedback**: Allows users to mark detected anomalies as false positives or user-aware anomalies. Imported feedback helps suppress matching anomalies from future inference results, reducing repeated noise and helping users focus on new or actionable findings.
- **Vendor and device agnostic**: Supports multi-vendor environments by applying the same detection process across different devices and operating systems.
- **Periodic scheduling and on-demand analysis**: Supports both scheduled and on-demand anomaly detection, enabling continuous compliance and faster troubleshooting.
- **Reduction in manual effort and human error**: Automates routine configuration audits and anomaly detection, minimizing the risk of oversight or misconfiguration due to manual processes.

Prerequisites

Download the CWAICTL CLI tool

All tasks related to managing logical groups and training the model are performed using CWAICTL, a command-line interface (CLI) tool. You must download CWAICTL to perform these operations.

1. Open your browser and navigate to:

   ```
   https://<cwai-host>/dist/
   ```

2. Download the package that matches your operating system:
   - cwaictl-linux-amd64.tar.gz – Linux (x86_64)
   - cwaictl-linux-arm64.tar.gz – Linux (ARM64)
   - cwaictl-darwin-amd64.tar.gz – macOS (Intel)
   - cwaictl-darwin-arm64.tar.gz – macOS (Apple Silicon)
   - cwaictl-windows-amd64.zip – Windows (x86_64)
3. Extract the archive. For example, on Linux (x86_64).

   ```
   tar -xzf cwaictl-linux-amd64.tar.gz
   ```

   The extracted archive contains:

   - cwaictl-linux-amd64 – The binary executable
   - README.md – Quick reference documentation
   - CWAICTL-USER-GUIDE.md – Detailed user guide
4. Copy the binary to a directory in your PATH or run directly.

   ```
   sudo cp cwaictl-linux-amd64 /usr/local/bin/cwaictl
   ```

   Alternatively, you can add the current directory to your PATH and create an alias:

   ```
   export PATH="$PATH:$(pwd)"
   alias cwaictl=cwaictl-linux-amd64
   ```

5. Verify the installation.

   ```
   cwaictl buildinfo
   ```

   This displays build information confirming the installation.

Configuration Drift Detection workflow

The Configuration Drift Detection workflow consists of these steps:

1. **Set up device and logical groups**: Group devices by similar characteristics, such as network roles or vendor types to form logical groups. When creating these groups, include as many device configurations as possible to improve accuracy. You can create logical groups manually or leverage NSO device groups, depending on your operational preference.
2. **Train the CLM model**: Train the CLM model using configurations from the selected logical group. The model autonomously learns configuration patterns, enabling it to detect anomalies during inference.
3. **Run inference to detect anomalies**: Analyze device configurations by running inference either on-demand or on a schedule. You can run inference by:
    - Selecting the logical group for analysis (manually created or NSO-mapped).
    - Uploading a CSV file containing device names for analysis.
    - Uploading new configuration files for analysis.
    - Setting up automatic, recurring analysis at defined intervals.
4. **Manage anomaly feedback**: Review detected anomalies and provide feedback for anomalies that do not require further action. You can export anomalies to a CSV file, mark anomalies as false positives or user-aware anomalies, and import the updated CSV file back into the CDD agent. Imported feedback helps suppress matching anomalies in future inference results for the same logical group.

Set up device and logical groups

To use Configuration Drift Detection, you need to organize your device configurations into logical groups. Logical groups are collections of device configurations that share similar characteristics, such as network roles or vendor types. Grouping devices ensures that the dataset contains uniform configuration patterns, enabling effective and accurate CLM model training. Mixing devices from different vendors or roles can result in inaccurate anomaly detection and outcomes.



If you have configurations from different vendors and network roles, you must create separate device groups. Each device must belong to only one group.

Anomaly feedback is associated with the logical group used during inference. After you import feedback for detected anomalies, matching anomalies are suppressed in future inference results for that logical group.

The model produces more accurate results when it is trained with sufficient representative configuration data. If the training dataset is too small, the results may be less accurate

Create logical groups using NSO device groups

1. **Define device groups in NSO**

   Create device groups in NSO by grouping multiple devices that share similar characteristics. Groups can include up to 50 devices in NSO. Use as many device configurations as possible to improve the model's accuracy.

   For instructions on creating device groups in NSO, see Create a device group.

2. **Identify device group**

   Once you have defined the required device groups in NSO, they are automatically mapped as logical groups in the CDD agent. A logical group has two status attributes:

   - **Training ready**: Enabled when a new group is created.
   - **Inference ready**: Enabled once the group has its first trained version.

   To identify logical group:

   a. List all logical groups mapped to the CDD agent:

   ```
   cwaictl run config-drift logical-groups get --all
   ```

   Parameters:

   - --all: Retrieves all existing logical groups.

   Example:

   ```
   cwaictl run config-drift logical-groups get --all
   Getting all logical groups ...
   ```

```
Using CA certificate from: /tmp/foresight-ca.crt
Duration: 0s
Group name: cisco-core
Attributes:
Training ready: true
Inference ready: false
Config start char: !
```

The output displays each group's name, attributes, and status flags indicating whether it is ready for training and inference.

In the example, the group is marked as "Training ready: true" but "Inference ready: false," which means it is ready for training but cannot yet be used for analysis since it has not been trained. There are no attributes because the group was created in NSO.

b. Identify the device group you want to use for training the CLM model.

Manually create a logical group

Manually create a logical device groups by grouping multiple devices that share similar configuration patterns.

Run this command to create a logical device group:

```
cwaictl run config-drift logical-groups create \
  --attrs <key=value> \
  --dir <config-directory> \
  [--config-start-char <char>]
```

Parameters:

- --attrs <key=value>: Defines the group's identifying attributes (metadata). These attributes are used to create the group name.
- --dir: Path to the directory containing configuration files that will be used for training.
- (optional) --config-start-char <char>: ASCII character indicating the start of a configuration file. In many Cisco device configs, the exclamation mark ! is used as a delimiter or comment line.

Example:

```
cwaictl run config-drift logical-groups create --attrs device=cisco,network=core --dir
~/configs --wait

Using CA certificate from: /tmp/foresight-ca.crt
Upload successful, ID: d3c625bc-1252-4b25-8125-ec705c46c381
Files uploaded successfully uploadID d3c625bc-1252-4b25-8125-ec705c46c381
Using CA certificate from: /tmp/foresight-ca.crt
```

```
Task ID: 2
Waiting for task 2 result...
Using CA certificate from: /tmp/foresight-ca.crt
Duration: 0s
Message: Group cisco_core created.
Group: cisco_core
Attributes:
 device: cisco
 network: core
Training ready: true
Inference ready: false
Config start char: !
Configuration files:
 File name: config_PE-1.cfg, Created at: 2025-10-24 07:24:12
 File name: config_PE-10.cfg, Created at: 2025-10-24 07:24:12
 File name: config_PE-100.cfg, Created at: 2025-10-24 07:24:12
...
```

Result:

Once the group is created, the output displays the group's name, attributes, and status flags indicating whether it is ready for training and inference.

In the example, the group is marked as "Training ready: true" but "Inference ready: false," which means it is ready for training but cannot yet be used for analysis since it has not been trained.

Train the model

The CLM model creates a configuration file for each device in the logical group, learning the existing patterns and relationships within those configurations.

1. To train the CLM model, run the command:

   ```
   cwaictl I want to run config-drift logical-groups train --group <logical group name>
   --wait
   ```

   Parameters:

   - --group <logical group name>: Specifies the logical group for model training.
   - --wait: Waits for 600 seconds for the training task to complete before returning control. If the training task is still running after this period, the command returns with no result. In that case, users can retrieve the training result by running the command cwaictl task get-result --task-id <ID>.

   Example:

   ```
   cwaictl I want to run config-drift logical-groups train --group cisco-core-grp --wait

   Using CA certificate from: /tmp/foresight-ca. crt
   Task ID: 26
   Waiting for task 26 result...
   Using CA certificate from: /tmp/foresight-ca. crt
   Message: Model training completed for group cisco-core-grp
   Group name: cisco-core-grp
   Inference ready: true
    Training ready: true
   Duration: 126s
   New model information:
    Version: 1
    Accuracy: 0.8980099558830261
    Loss: 1.6444802284240723
    Configuration files:
     File name: config_PE-I.cfg
     Device name:
     Created at: 2025-10-24 07:24:12
     File name: config_PE-10. cfg
     Device name:
     Created at: 2025-10-24 07:24:12
     File name: config_PE-100.cfg
     Device name:
     Created at: 2025-10-24 07:24:12
   ```

```
...
```

Result:

- The output displays the training status, model version, accuracy, loss, and details of configuration files used in training.
- Configuration files: Represent the repository of files used to train the model.

2. To check the training status, run the command:

```
cwaictl I want to run config-drift logical-groups get --group <logical group name>
```

Example:

```
cwaictl I want to run config-drift logical-groups get -group cisco-core-grp

Getting information for logical group cisco-core-grp
Using CA certificate from: /tmp/foresight-ca.crt
Duration: 0s
Group name: cisco-core-grp
Attributes:
 device: cisco
 network: core
Training ready: true
Inference ready: true
Config Start Char: !
Default model:
 Version: 1
 Status: completed
 Accuracy: 0.8980099558830261
 Loss: 1.6444802284240723
 Train date: 2025-10-24 07:24:12
 Repository:
  File name: config_PE-I.cfg
  Device name:
  Created at: 2025-10-24 07:24:12
  File name: config_PE-10.cfg
  Device name:
  Created at: 2025-10-24 07:24:12
  File name: config_PE-100.cfg
  Device name:
  Created at: 2025-10-24 07:24:12
...
```

Result:

- The output shows a "Default model" section that provides details about the model version, training status, accuracy, loss, and training date.
- Version: Indicates the version number of the trained model for this logical group. Each time you retrain the model (for example, after adding or updating configuration files), a new version is created.

Run inference to detect configuration anomalies

This section explains how to use the Configuration Drift Detection agent to analyze device configurations and detect anomalies.

Access Configuration Drift Detection agent

To launch Crosswork AI in NSO, click the Launch AI Assistant icon .

The Crosswork AI user interface opens.

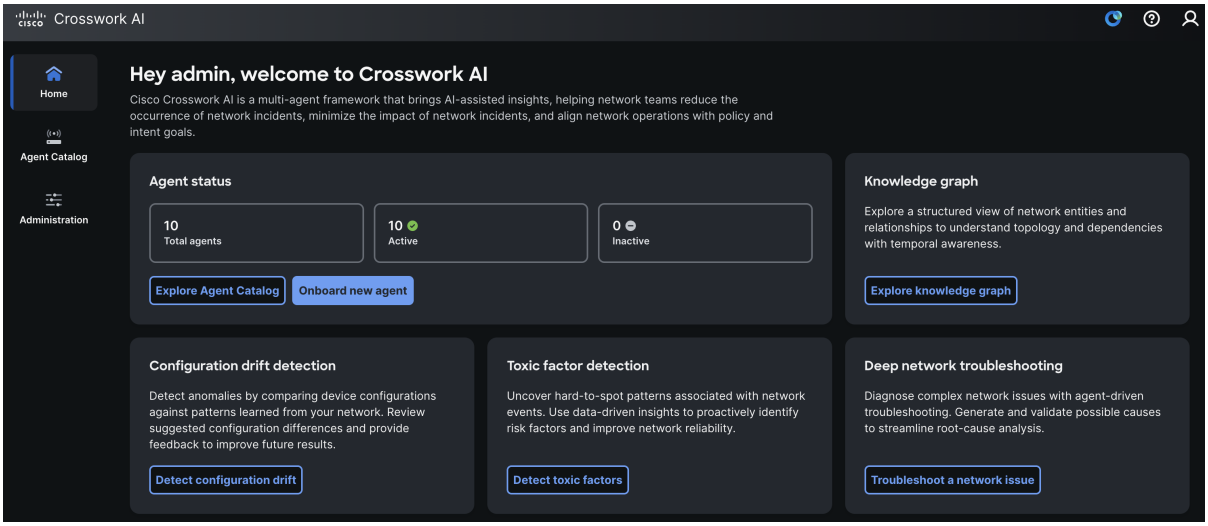


Figure 1. Crosswork AI home page

Crosswork AI provides pre-defined suggestion tiles for interacting with the agent. All actions performed through the AI assistant are specific to your user account.

You can access the Configuration Drift Detection agent in one of the following ways:

- From the Crosswork AI home page, select **Detect configuration drift**.
- Click to open AI Assistant. Select **I want to run configuration drift detection**, and then click [Send icon] to send the request.

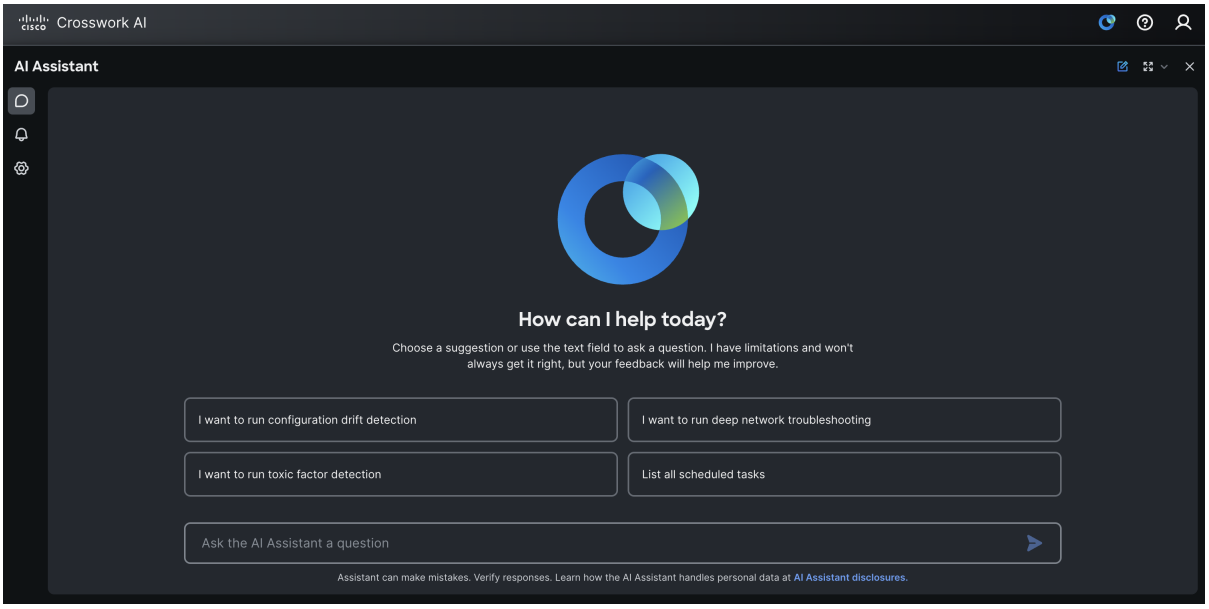


Figure 2. Crosswork AI Assistant page

The Configuration Drift Detection provides different methods to run inference tasks on-demand or schedule them to run automatically. When running inference on-demand, you can either select a logical group for analysis or upload a list of specific device names. Alternatively, you can upload new device configurations for analysis or schedule inference tasks to run at set intervals.

After inference is complete, you can review detected anomalies and provide feedback for anomalies that do not require further action. You can mark anomalies as false positives or user-aware anomalies by exporting the anomaly feedback CSV, updating the feedback fields, and importing the CSV back into the Configuration Drift Detection.

Choose the method that best fits your operational workflow.

- Analyze configurations on-demand for specific device names
- Analyze configurations on-demand for a logical group
- Analyze new configurations on-demand
- Schedule inference to run later
- Manage anomaly feedback

Analyze configurations on-demand for specific device names

To analyze configurations on a list of specific device names:

1. In the Configuration Drift Detection, select **On-demand**.

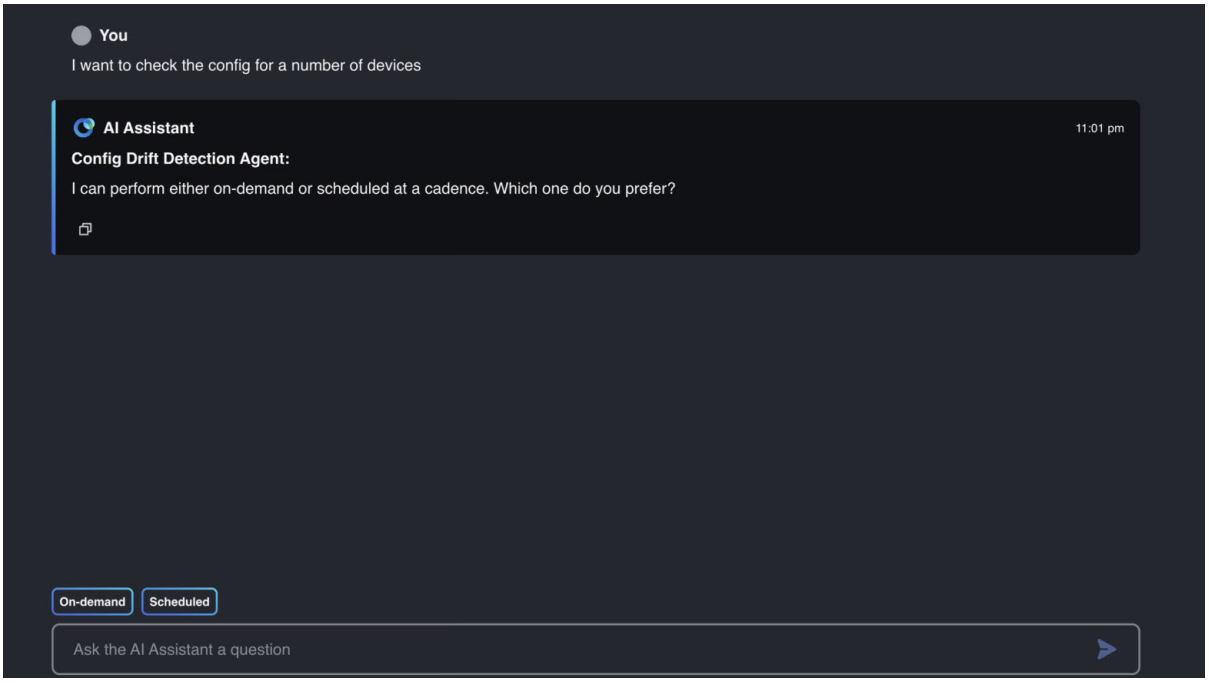


Figure 3. Configuration Drift Detection agent

2. Select **Fetch automatically** and then choose **I want to upload device names**.

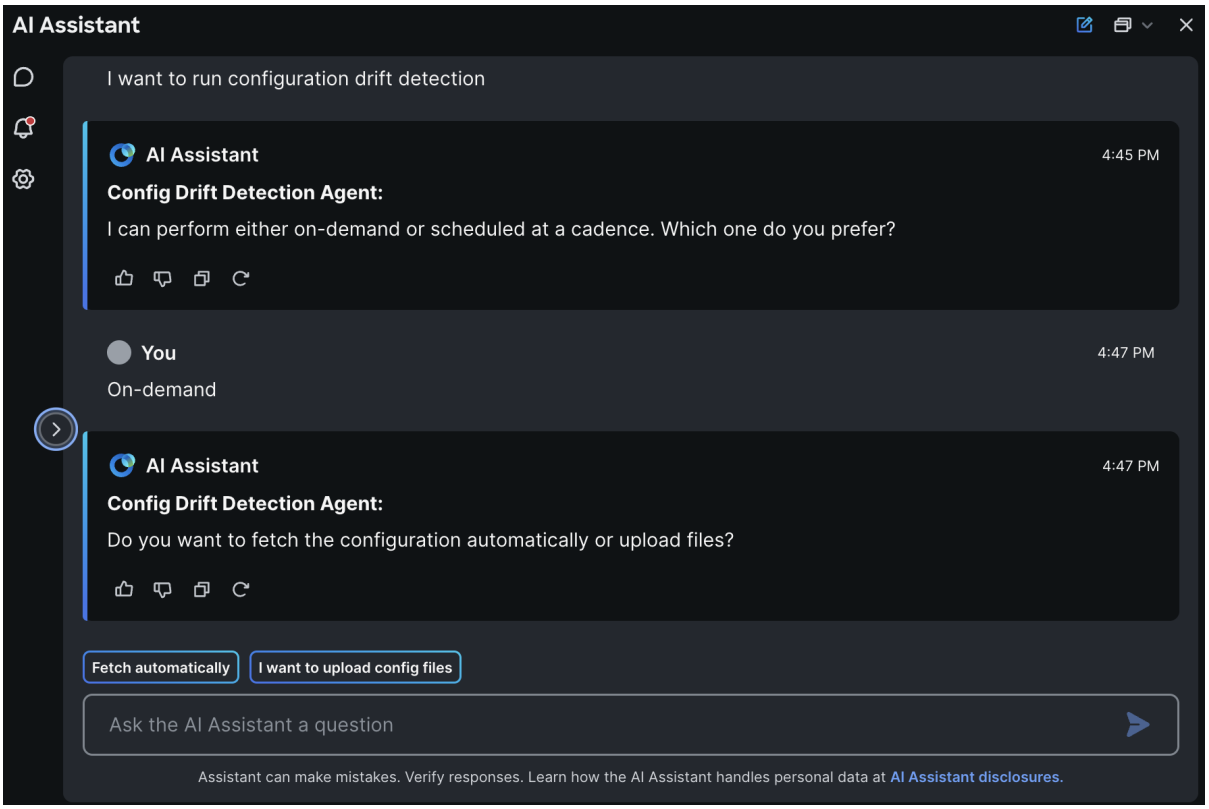


Figure 4. CDD - Fetch automatically

3. Click **Upload files** to select a CSV file containing the list of device names you want to analyze. The file must follow RFC 4180 format, contain a single column, and list device names on separate rows.

   Note: Devices must belong to a trained device group. The Configuration Drift Detection only performs analysis if the corresponding device group has already been trained.

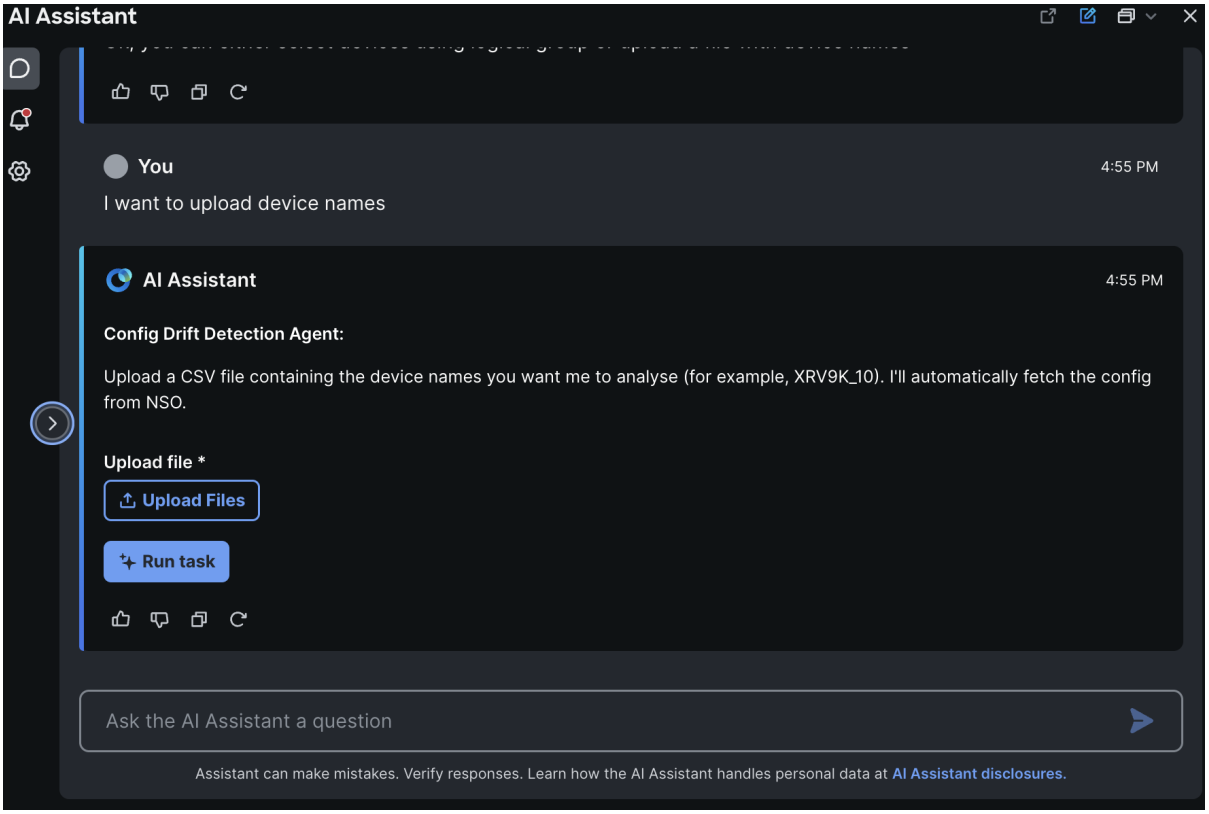


Figure 5. CDD - Upload device names

4. Select **Run task**.

   The agent determines the device group for each device, retrieves the configurations from NSO, and runs analysis. Each device is processed sequentially. Once analysis is complete for one device, the next device is processed. After completion, the agent displays the results of its findings.

   In the example, the agent analyzed three devices and found 20 anomalies. One device could not be matched with a trained device group and was excluded from the analysis.

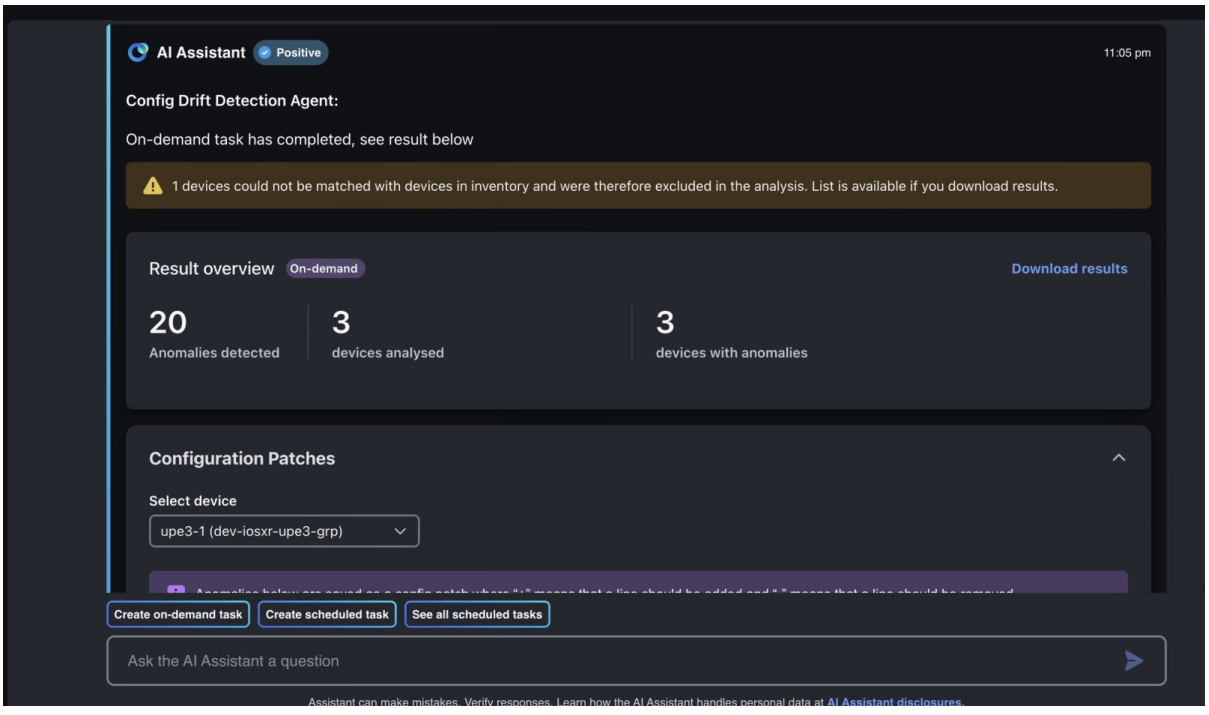


Figure 6. CDD inference result

5. Click the **Download results** link to download the results in JSON format.
6. To view anomalies for a specific device, choose the device from the **Select device** list. The agent displays all detected anomalies, using a " +" sign to indicate lines that should be added and a " -" sign to indicate lines that should be removed.
7. To provide feedback on detected anomalies, export the anomaly feedback CSV, update the feedback fields, and import the CSV back into the Configuration Drift Detection. For more information, see Manage anomaly feedback.

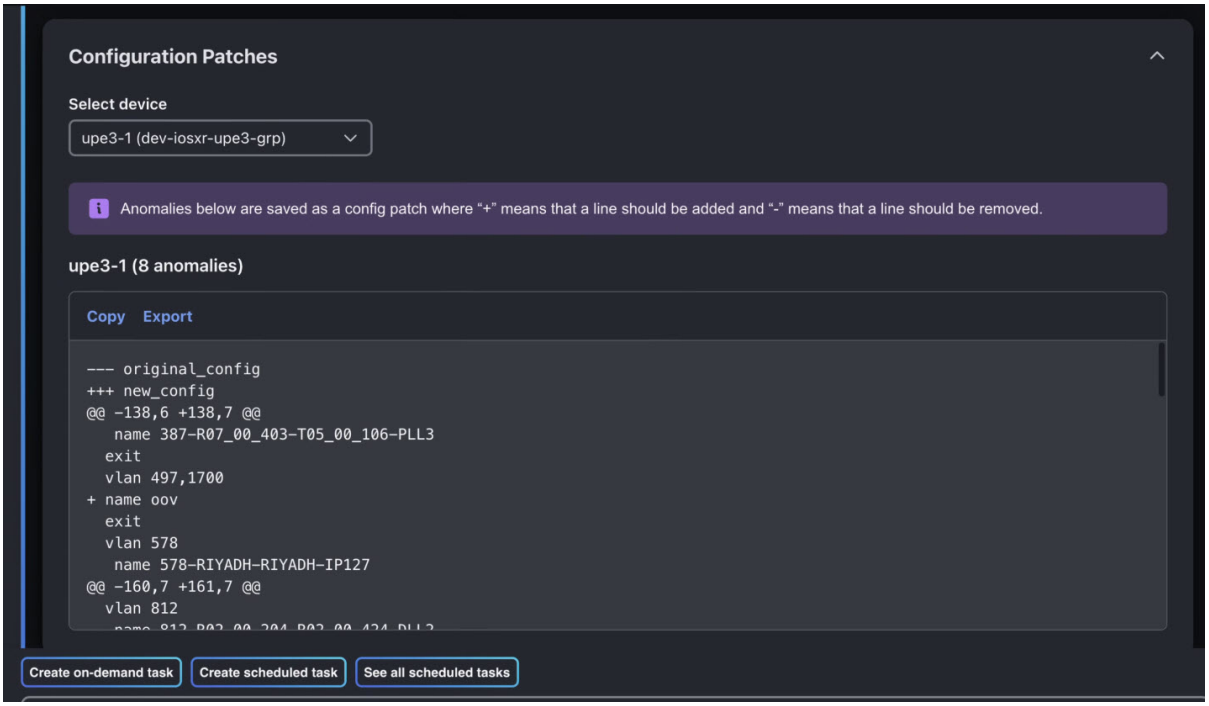


Figure 7. CDD inference anomalies

8. Use the **Copy** link to copy the configuration patch to your clipboard for easy sharing or reporting.
9. Use the **Export** link to download the configuration patch in JSON format.

Additional options

- **Create on-demand task**: To run another on-demand inference task.
- **Create scheduled task**: To set up automated inference tasks that run at regular intervals.
- **See all scheduled tasks**: Displays all existing scheduled tasks.

Analyze configurations on-demand for a logical group

To analyze configurations for a specific logical group:

1. In the Configuration Drift Detection, select **On-demand**, then choose **Fetch from NSO**.
2. Select **Use logical groups**.

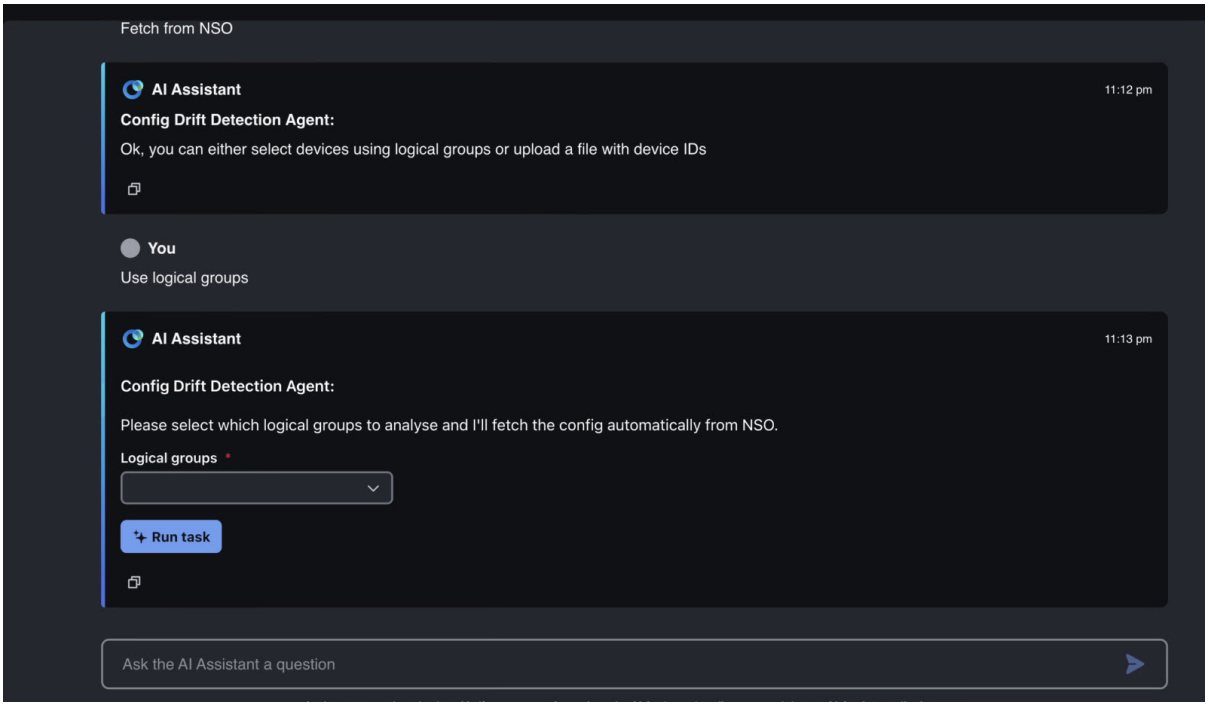


Figure 8. CDD – Use logical groups

3. From the **Logical groups** list, select a device group for analysis.
4. Select **Run task**.

   The agent sequentially performs inference on every device in the group and summarizes the results for all configurations.

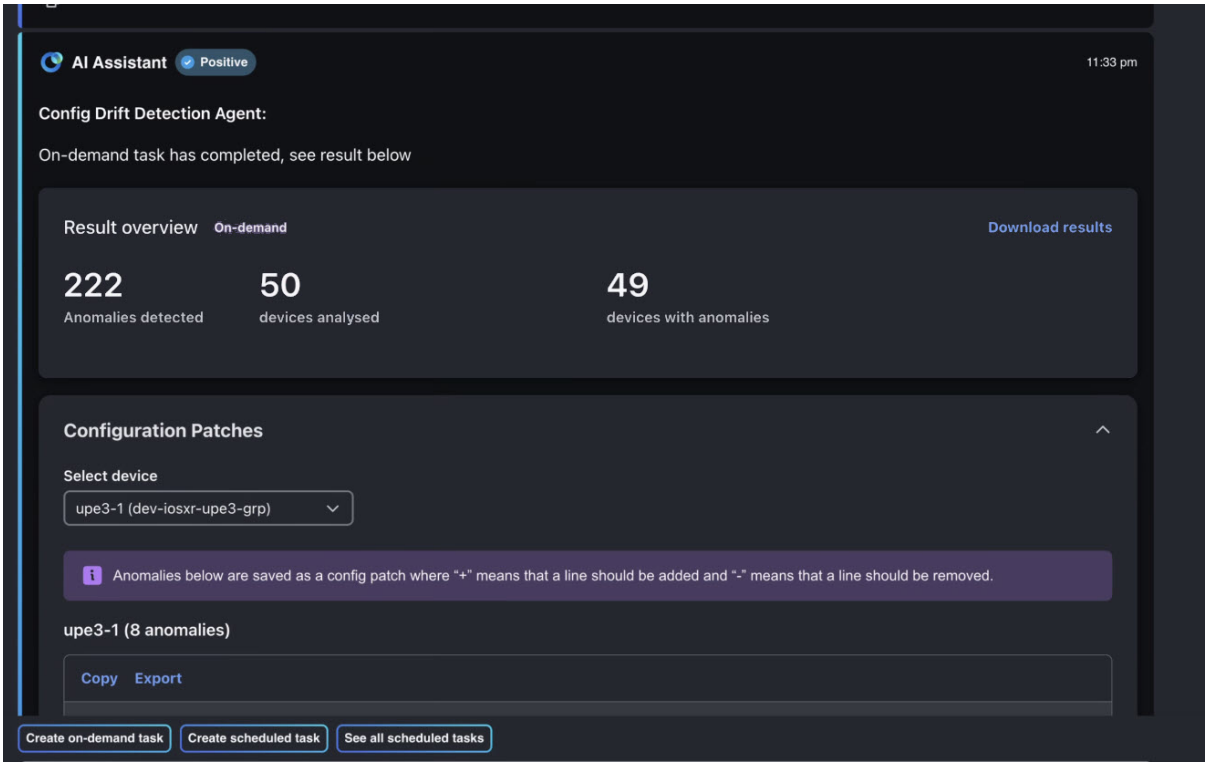


Figure 9. CDD inference result

5. Click the **Download results** link to download the results in JSON format.
6. To view anomalies for a specific device, choose the device from the **Select device** list. The agent displays all detected anomalies, using a " +" sign to indicate lines that should be added and a " -"

sign to indicate lines that should be removed.

7. Use the **Copy** link to copy the configuration patch to your clipboard for easy sharing or reporting.
8. Use the **Export** link to download the configuration patch in JSON format.

Additional options

- **Create on-demand task**: To run another on-demand inference task.
- **Create scheduled task**: To schedule automated inference tasks to run at regular intervals.
- **See all scheduled tasks**: Displays all existing scheduled tasks.

Analyze new configurations on-demand

To analyze new configuration files:

1. In the Configuration Drift Detection, select **On-demand**, then choose **I want to upload config files**.
2. From the **Logical groups** list, select a trained group you want to test the files against.
3. Click **Upload files** to select one or more configuration files for analysis.

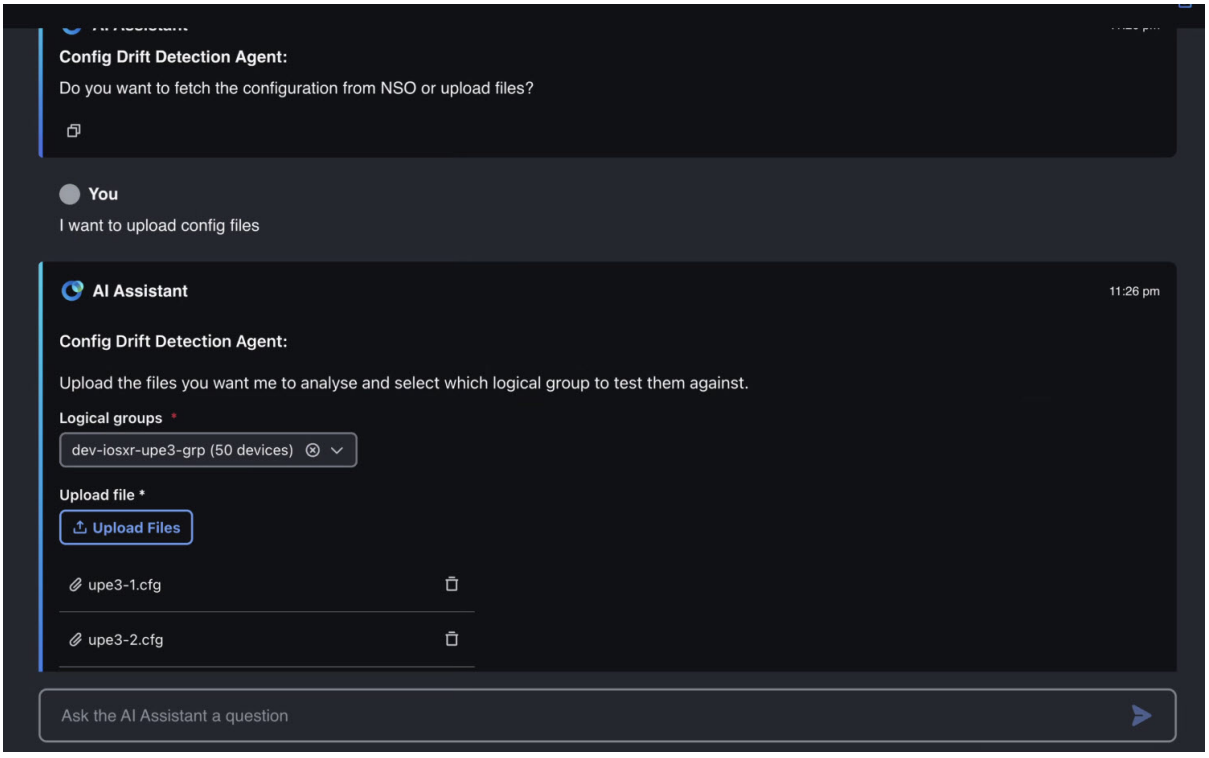


Figure 10. CDD - Upload config files

4. Select **Run task**.

   The agent analyzes each configuration file against the selected trained logical group and displays the results.

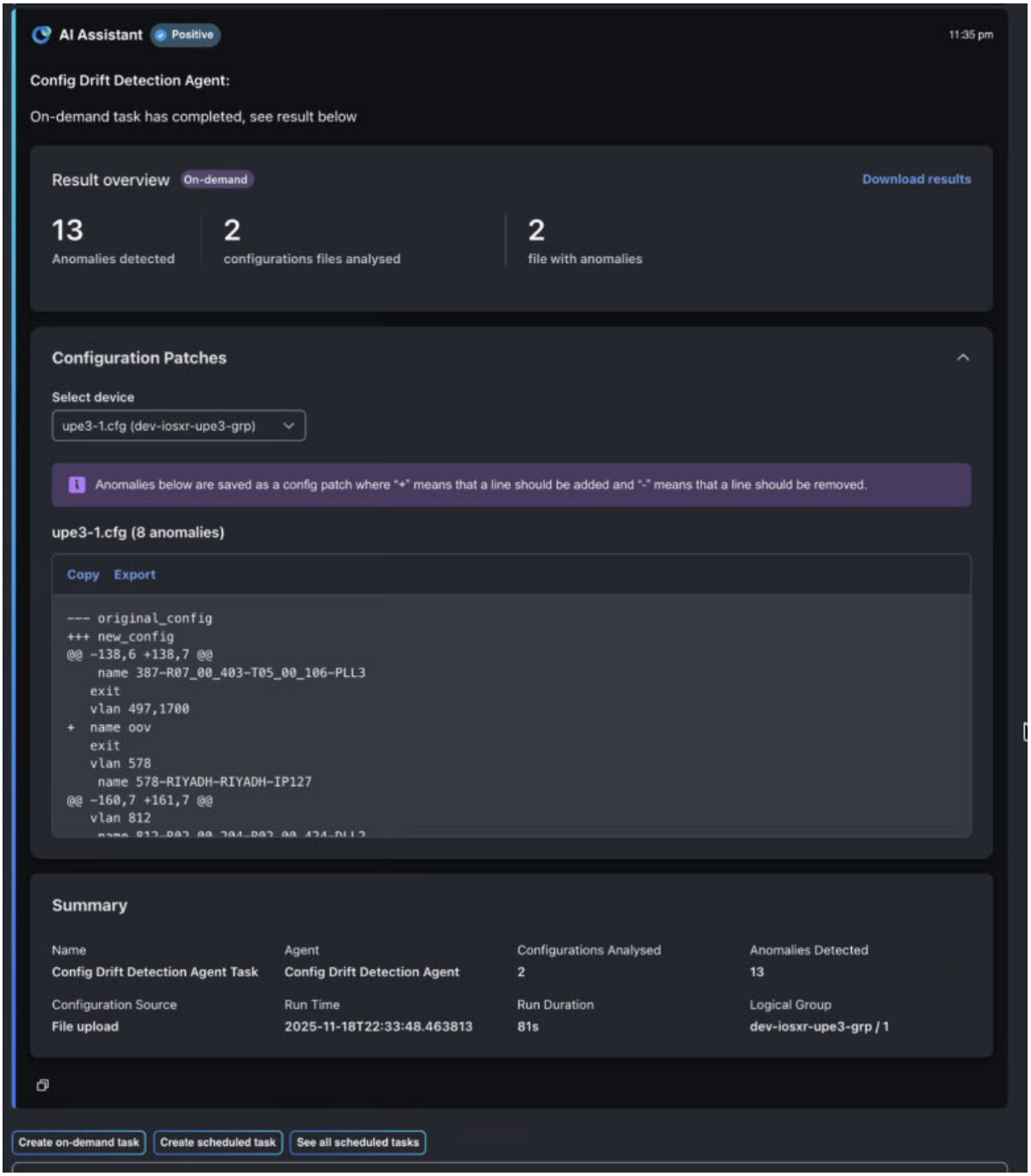


Figure 11. CDD inference result

5. Click the **Download results** link to download the results in JSON format.
6. To view anomalies for a specific device, choose the device from the **Select device** list. The agent displays all detected anomalies, using a " +" sign to indicate lines that should be added and a " -" sign to indicate lines that should be removed.
7. Use the **Copy** link to copy the configuration patch to your clipboard for easy sharing or reporting.
8. Use the **Export** link to download the configuration patch in JSON format.

Additional options

- **Create on-demand task**: To run another on-demand inference task.
- **Create scheduled task**: To schedule automated inference tasks to run at regular intervals.
- **See all scheduled tasks**: Displays all existing scheduled tasks.

Schedule inference to run later

Scheduled inference provides continuous monitoring, helping maintain network stability by proactively identifying configuration drift.

To automate the inference process by setting a schedule:

1. In the Configuration Drift Detection, select **Scheduled**.

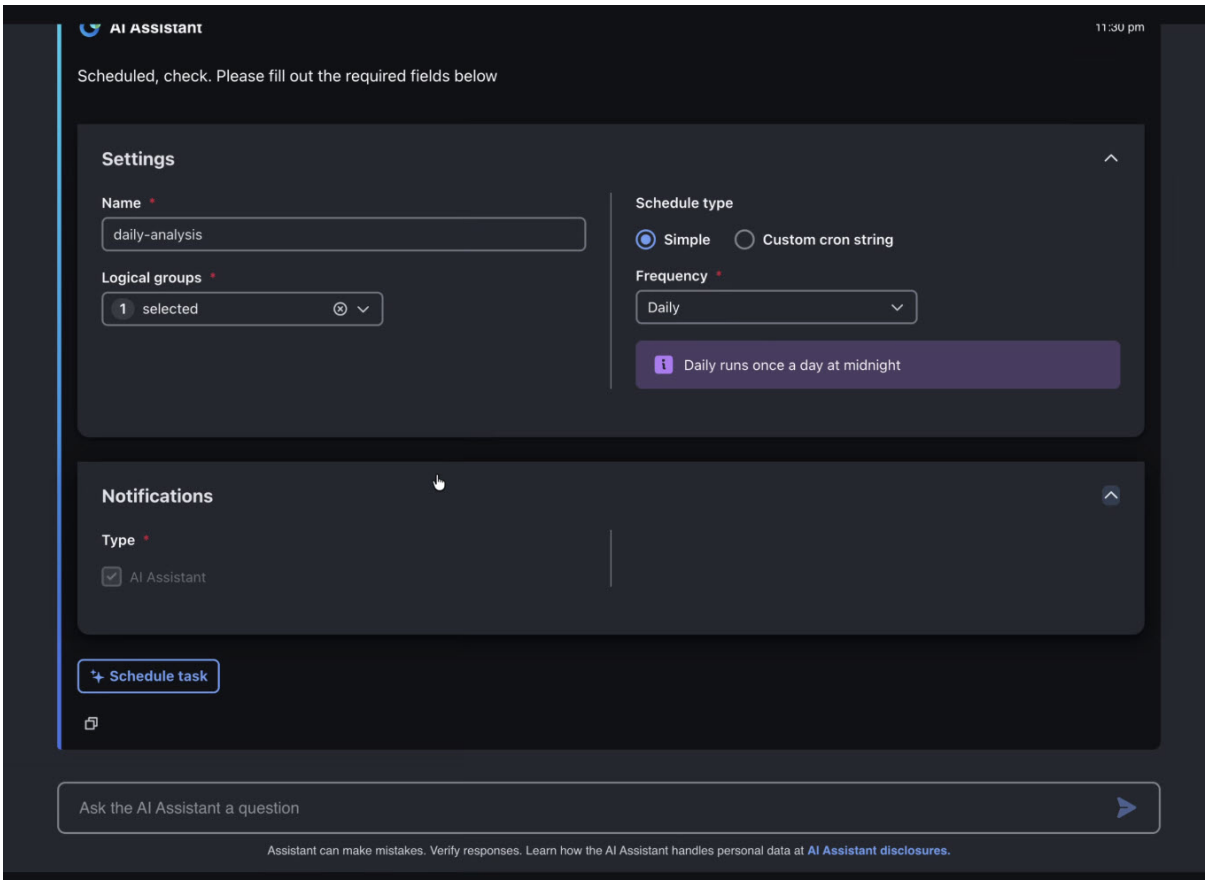


Figure 12. CDD schedule task settings

2. Enter a name for the schedule.
3. From the **Logical groups** list, select the group you want to analyze.
4. For the **Schedule type**, choose how you want to set when the task runs.
    - **Simple**: Select this option to pick a standard schedule from the **Frequency** list, such as daily, weekly, or monthly.
    - **Custom cron string**: Select this option to enter a more advanced schedule using a cron expression. This allows you to specify exactly when the task should run. For example, entering 0 2 * * 1 would schedule the task to run every Monday at 2:00 AM. For help with generating cron expressions, you can use online tools such as Cronhub.
5. Select **Schedule task**. By default, AI Assistant notifications are enabled.

    Based on the configured schedule and settings, the agent processes each device in the logical group sequentially, fetching its configuration and running inference at the specified intervals.

    Once a scheduled task run is complete, you will receive a notification. To review the results, click the icon in the left menu, then select the specific scheduled task.

Additional options

- **See all scheduled tasks**: Displays all existing scheduled tasks. For each scheduled task, you can also:
  - Suspend a task to temporarily stop it from running.
  - Delete a task to permanently remove it from the schedule.
  - Activate a suspended task to resume its execution according to the defined schedule.
- **List recent runs for this task**: Shows up to 10 of the most recent runs for the task. Click a run to view its results.

Manage anomaly feedback

Anomaly feedback allows you to mark detected anomalies that do not require further action. You can use anomaly feedback to reduce repeated noise in future inference results for the same logical group.

You can mark an anomaly as one of the following:

- **False positive**: The anomaly is not a valid issue and does not need to be shown again.
- **User-aware anomaly**: The anomaly is known, expected, or already being handled and does not require further action.

After you import anomaly feedback, matching anomalies are suppressed in future inference results for the logical group.



Anomaly feedback does not replace model training. The model still detects anomalies during inference. Feedback is applied to suppress matching anomalies from the displayed results.

Download anomaly feedback CSV

After an inference task is complete, download the anomaly feedback CSV from the inference results.

The CSV contains the detected anomalies for the task. If the task includes multiple devices, the anomalies are included in a single CSV file.

Use this CSV file to mark anomalies as false positives or user-aware anomalies.

Update the anomaly feedback CSV

Open the downloaded CSV file and update the feedback fields for the anomalies you want to mark.

The CSV includes information such as the task ID, device name, logical group, anomaly details, feedback comment, false positive flag, and user-aware anomaly flag.

To mark an anomaly as a false positive, update the false positive field for that row.

To mark an anomaly as user-aware, update the user-aware anomaly field for that row.

The feedback comment field is optional.

Do not mark the same anomaly as both a false positive and a user-aware anomaly.

Import anomaly feedback CSV

After you update the CSV file, import it back into the Configuration Drift Detection.

The Configuration Drift Detection validates the uploaded CSV and imports valid feedback rows. After the feedback is imported, matching anomalies are suppressed in future inference results for the same logical group.

Verify anomaly feedback

To verify that anomaly feedback was applied:

1. Run inference again on the same logical group.
2. Review the anomaly results.
3. Confirm that the marked anomaly is no longer displayed.

If a matching anomaly appears on another device in the same logical group, the Configuration Drift Detection suppresses the matching anomaly based on the imported feedback.

Reverse anomaly feedback

You can reverse previously imported anomaly feedback if an anomaly was marked incorrectly.

To reverse anomaly feedback:

1. Download the anomaly feedback CSV again.
2. Locate the anomaly that was previously marked.
3. Remove or update the feedback flag.
4. Import the updated CSV back into the Configuration Drift Detection.
5. Run inference again.

After the feedback is reversed, the anomaly can appear again in future inference results if the Configuration Drift Detection detects it.

Anomaly feedback behavior and limitations

Consider the following when using anomaly feedback:

- Anomaly feedback applies to future inference results for the logical group.
- Feedback suppresses matching anomalies from displayed results.
- Feedback does not retrain the model or replace the training workflow.
- You can reverse feedback if an anomaly was marked incorrectly.
- Feedback entries may be retained only for a configured period before they are purged.

Interpret anomaly detection

Types of anomalies

The CDD agent is designed to identify a range of configuration anomalies that may indicate deviations from the expected configuration. These include:

Parameter changes: Detection of values or settings that have been altered from the expected configuration.

Missing configuration lines or blocks: Identification of lines or sections that are absent from the configuration but are expected based on the learned configuration patterns.

Unexpected configuration lines or blocks: Detection of additional lines or sections that should not be present according to what is expected.

Multiple or consecutive anomalies: Recognition of scenarios where anomalies appear close together.

After reviewing detected anomalies, you can provide feedback for anomalies that do not require further action. Mark an anomaly as a false positive if it is not a valid issue, or as a user-aware anomaly if it is known, expected, or already being handled. For more information, see Manage anomaly feedback.

Interpreting results

The CDD agent presents anomalies in a diff-style format, making it easy to spot differences between the current configuration and the expected configuration:

- **Lines beginning with +**

  Indicate configuration items that are missing from your device's configuration and are recommended to be **added**.

  Example:

  ```
  + load-interval 300
  ```

- **Lines beginning with -**

  Indicate configuration items that are unexpected or should be **removed** because they do not match what is expected.

  Example:

  ```
  - spanning-tree bpdufilter enable
  ```

This format helps you quickly identify and address configuration differences. You can review each flagged line to determine if the change is intentional or requires remediation, and use the suggested additions and deletions to bring device configurations back in line with your network's standards.

Agent management with CWAICTL

Logical groups

- **List all groups**

```
cwaictl I want to run config-drift logical-groups get --all
```

- **Get a specific group information**

```
cwaictl I want to run config-drift logical-groups get --group <group-name>
```

- **Get a specific model version**

```
cwaictl I want to run config-drift logical-groups get --group <group-name> --version
<version>
```

- **Create a logical group**

```
cwaictl I want to run config-drift logical-groups create \
  --attrs <key=value> \
  --dir <config-directory> \
  [--config-start-char <char>]
```

Example:

```
cwaictl I want to run config-drift logical-groups create \
  --attrs vendor=cisco,device_type=switch \
  --dir ./cisco-configs \
  --config-start-char "!"
```

Parameters:

  - `--attrs <key=value>`: Defines the group's identifying attributes (metadata).
  - `--dir`: Path to the directory containing configuration files.
  - (optional) `--config-start-char <char>`: ASCII character indicating the start of a configuration file. In many Cisco device configs, the exclamation mark ! is used as a delimiter or comment line.

Add configuration files

```
cwaictl I want to run config-drift logical-groups add \
  --group <group-name> \
  --dir <config-directory> \
  [--policy <merge|replace-all|merge-with-overwrite>]
```

- **Merge new configs**

  Example:

  ```
  cwaictl I want to run config-drift logical-groups add \
    --group cisco_edge \
    --dir ./new-configs \
    --policy merge
  ```

- **Replace all configs**

  Example:

  ```
  cwaictl I want to run config-drift logical-groups add \
    --group cisco_edge \
    --dir ./updated-configs \
    --policy replace-all
  ```

Parameters:

- `--group <group-name>`: The name of the existing logical group to which you want to add new configuration files.
- `--dir <config-directory>`: Path to the directory containing the configuration files you want to add to the group.
- `--policy <merge|replace-all|merge-with-overwrite>`: Defines how the new configs should be handled when adding them to the group.

Train the model

```
cwaictl I want to run config-drift logical-groups train \
  --group <group-name> \
  [--wait]
```

Example:

```
cwaictl I want to run config-drift logical-groups train \
```

```
  --group cisco_edge \
  --wait
```

Parameters:

- `--group <group-name>`: The name of the logical group whose configs will be used to train the model.
- (optional) `--wait`: Waits until the task is complete before returning results.

Analyze configurations by running inference tasks

```
cwaictl I want to run config-drift inference \
  [--file <config-file>] \
  --group <group-name> \
  [--devices-list <csv-file>] \
  [--wait]
```

- **Analyze single configuration file**

  Example:

  ```
  cwaictl I want to run config-drift inference \
    --file ./device-config.txt \
    --group cisco_edge \
    --wait
  ```

- **Analyze against multiple groups**

  Example:

  ```
  cwaictl I want to run config-drift inference \
    --file ./unknown-config.txt \
    --group cisco_edge \
    --group juniper_core \
    --group arista_access \
    --wait
  ```

- **Analyze a list of device IDs**

  Example:

  ```
  cwaictl I want to run config-drift inference \
    --devices-list ./devices.csv \
    --group cisco_edge \
  ```

```
--wait
```

Parameters:

- --file <config-file>: Path of the config file to analyze against trained groups.
- --group <group-name>: Trained logical group name.
- --devices-list <csv-file>: Path to a CSV file listing multiple devices to analyze. The file must follow RFC 4180 format, contain a single column, and list device names on separate rows.
- (optional) --wait: Waits until the task is complete before returning results.

Delete group

- **Delete entire group**

```
cwaictl I want to run config-drift logical-groups delete --group <group-name>
```

- **Delete specific version**

```
cwaictl I want to run config-drift logical-groups delete --group <group-name> --version <ver>
```

Parameters:

- --group <group-name>: The name of the logical group you want to delete.
- --version <ver>: Deletes a specific model version within the logical group instead of removing the entire group.



By default, Crosswork Network Controller deletes events after 30 days and deletes active and cleared device alarms after 60 days. If your deployment requires this data for a longer period, review and update the CNC cleanup settings before the applicable retention period expires.

Reset group repository

```
cwaictl I want to run config-drift logical-groups reset --group <group-name>
```

Parameters:

- --group <group-name>: The name of the logical group whose repository you want to reset. It clears all configuration files and associated data from the group while keeping the group intact.

Troubleshooting Configuration Drift Detection agent

Error: " No groups found"

If you receive an error like " No groups found." , check direct connectivity to NSO:

```
curl -u <nso-user>:<nso-pass> \
  -H " Accept: application/yang-data+json"  \
  http://<nso-host>:8080/restconf/data/tailf-ncs:devices/device-group
```

Training job times out

If model training times out, it may be due to a large number of configurations or insufficient resources. Inference can also take a long time for various reasons, such as analyzing the entire device group that contains many devices.



The options below are intended for advanced users. Changing these settings can affect system performance and stability. If you are unsure about making these changes, contact Cisco Support for assistance.

- **Increase global task timeout**

  If you are running a training job with the CDD agent and it times out (the default global task timeout is 1 hour), it will be report as pod <POD_NAME> failed. You can extend the timeout by setting the FORESIGHT_AGENT_TIMEOUT variable.

  ```
  $ ./bin/cwaictl config set env foresight FORESIGHT_AGENT_TIMEOUT 7200
  Set 1 environment variables for deployment 'foresight'.
  Message: Environment variables updated successfully
  Rollout status: completed
  ```

  Verify that the variable is set:

  ```
  $ ./bin/cwaictl config get env foresight | grep FORESIGHT_AGENT_TIMEOUT
    FORESIGHT_AGENT_TIMEOUT=7200
  ```

- **Inference timeouts**

  Inference timeouts apply only to inference tasks. In the agent's settings, there are two timeout parameters that prevents the config-drift inference from running indefinitely in cases such as:

  - The overall dataset is very large and takes too long to process.

- A single device configuration is too complex or large.
- The agent hangs during processing.

Default value for both parameters is 0 (disabled), meaning timeouts are not enforced unless explicitly enabled in the task settings.

```
" settings" : {
   " per_task_timeout_seconds" : 0,
   " per_device_timeout_seconds" : 0
}
```

- per_task_timeout_seconds: Maximum allowed time for the entire inference task (processing all devices).
- per_device_timeout_seconds: Maximum allowed time for processing a single device configuration.

You can modify these timeout settings using the cwaictl command-line tool.

```
cwaictl agent settings set --name config-drift '{" per_device_timeout_seconds" : 3600,
" per_task_timeout_seconds" : 600}'
```

- **Training is slow or stuck: adjusting resource limits**

If the agent is slower than expected, it may reach a timeout. Its performance is driven by CPU limit setting. By default, the agent can use up to 1 core (1000m). You can increase the CPU allocation by updating the cpu_limit in the agent's metadata.

a. Check current resource settings.

```
$ ./bin/cwaictl agent get --name config-drift
{
  " name" : " config-drift" ,
  " full_name" : " Config Drift Detection Agent" ,
  " description" : " Detects configuration drift in network devices" ,
  " image" : " apps/cisco-crosswork-ai/foresight-config-drift" ,
  " version" : " main-266" ,
  " type" : " BUILTIN" ,
  " capabilities" : [],
  " metadata" : {
    " cpu_limit" : " 1000m" ,
    " cpu_request" : " 500m" ,
    " description" : " Detects configuration drift in network devices" ,
    " memory_limit" : " 2Gi" ,
    " storage_size" : " 100Mi" ,
    " memory_request" : " 1Gi" ,
```

```
    "tags" : [
      "config-drift"
    ],
    "timeout" : "600"
  },
  "settings" : {
    "per_task_timeout_seconds" : 0,
    "per_device_timeout_seconds" : 0,
    "token_variability_threshold" : 1000000000
  }
}
```


b. Look for cpu_limit and cpu_request under metadata. In the example, cpu_limit and cpu_request is set to 1000m and 500m respectively.

c. Export the agent's configuration to a template file.

```
./bin/cwaictl agent template --from-server config-drift --output
/tmp/config_drift.json
Template written to: /tmp/config_drift.json
  Source: server (config-drift)
  Note: Exported from server, ready to customize
```

d. Edit /tmp/config_drift.json and update the values for higher CPU allocation. In the example, cpu_limit and cpu_request is increased to 4000m and 2000m respectively.


```
...
  "metadata" : {
    "cpu_limit" : "4000m",
    "cpu_request" : "2000m",
    "description" : "Detects configuration drift in network devices",
    "memory_limit" : "2Gi",
    "memory_request" : "1Gi",
    "storage_size" : "100Mi",
    "tags" : [
      "config-drift"
    ],
    "timeout" : "600"
  },
...
```


e. Update the agent with the new configuration:

```
./bin/cwaictl agent update --name config-drift --from-file /tmp/config_drift.json
```

```
Agent 'config-drift' updated successfully
```