

CloudVision Mastery Workshop Lab Guide



ACTIONS

Actions can be used within a change control to run a series of commands on devices. There are a number of built-in actions that can be selected when configuring a change control operation or creating a change control template. You can also create custom actions by creating arguments and writing a Python 3 script to manipulate the arguments.

When configuring a change control, you'll select the dynamic values of an argument, like the device the action will run against and the image to install. When creating a custom action, you'll use dynamic arguments, like DeviceID, to enable users to select values when configuring a change control.

Create an Action

1. Navigate to **Provisioning > Actions**.
2. To create a new action, click on the “**+ New Action**” button at the top of the screen.

Provisioning

Network Provisioning

Configlets

Image Repository

Upload and Download...

Tasks

Actions

Change Control

Action Bundles

Templates

Studios

Workspaces

Snapshot Configuration

Public Cloud Accounts

Tags

Zero Touch Provisioning

Actions

Create and manage actions for use in action bundles and change control operations

Search actions

Custom Actions

Bounce Interface
Change Control
1 minute ago

Generate Campus Node IDs
Studio Autofill | Autofill Campus Studio Node IDs
1 month ago

Generate Data Centers Node IDs
Studio Autofill | Autofill L3LS Studio Node IDs
1 month ago

Port Auto Description
Change Control
1 minute ago

Built-In Actions

Capture CLI Snapshot
Change Control Built-in | Run a snapshot that has been defined in Snapshot Configuration.

Clean Flash
Change Control Built-in | Delete device files using a file spec and file glob. This works by collecting the existing files on the de...

Enter BGP Maintenance Mode
Change Control Built-in | Pair this action with Exit BGP Maintenance Mode to run specific tests detailed in the EOS User Manua...

Execute Task

Restart TerminAttr
Change Control
1 minute ago

Check MLAG Health
Change Control | Checks that a device's MLAG ports come up in a timely manner
1 minute ago

Download File
Change Control Built-in | Download image and extension files to a device flash directory. The files specified must be available I...

Enter ZTP
Change Control Built-in | The Enter ZTP action deletes the startup-config, zerotouch-config and reloads the device to force th...

Exit BGP Maintenance Mode

3. A **“New Action”** pop-up will appear. In the **“Name”** field, type your initials followed by **“- Clear Counters.”**
4. Click the blue **“Save”** button to save the new action.

Provisioning

Network Provisioning

Configlets

Image Repository

Upload and Download...

Tasks

Actions

Change Control

Action Bundles

Templates

Studios

Workspaces

Snapshot Configuration

Public Cloud Accounts

Tags

Zero Touch Provisioning

Actions

Create and manage actions for use in action bundles and ch

Search actions

Custom Actions

Bounce Interface
Change Control
20 hours ago

Generate Data Centers Node IDs
Studio Autofill | Autofill L3LS Studio Node IDs
1 month ago

Port Auto Description
Change Control
20 hours ago

Built-In Actions

Capture CLI Snapshot
Change Control Built-in | Run a snapshot that has been defined in Snapshot Configuration.

Clean Flash
Change Control Built-in | Delete device files using a file spec and file glob. This works by collecting the existing files on the de...

Enter BGP Maintenance Mode
Change Control Built-in | Pair this action with Exit BGP Maintenance Mode to run specific tests detailed in the EOS User Manua...

Execute Task
Change Control Built-in | Run this action with a pre-defined TaskID to execute the specified network changes.

Exit ZTP
Change Control Built-in | The Exit ZTP action copies the running config to cvp-config, the zero-touch agent will reboot the dev...

Restart TerminAttr
Change Control
20 hours ago

Check MLAG Health
Change Control | Checks that a device's MLAG ports come up in a timely manner
20 hours ago

Download File
Change Control Built-in | Download image and extension files to a device flash directory. The files specified must be available I...

Enter ZTP
Change Control Built-in | The Enter ZTP action deletes the startup-config, zerotouch-config and reloads the device to force th...

Exit BGP Maintenance Mode
Change Control Built-in | Pair this action with Enter BGP Maintenance Mode to run specific tests detailed in the EOS User Man...

Interface Cable Test
Change Control Built-in | Run this action to invoke the EOS I1 cable test feature. This diagnostic is useful to determine if a Bas...

New Action

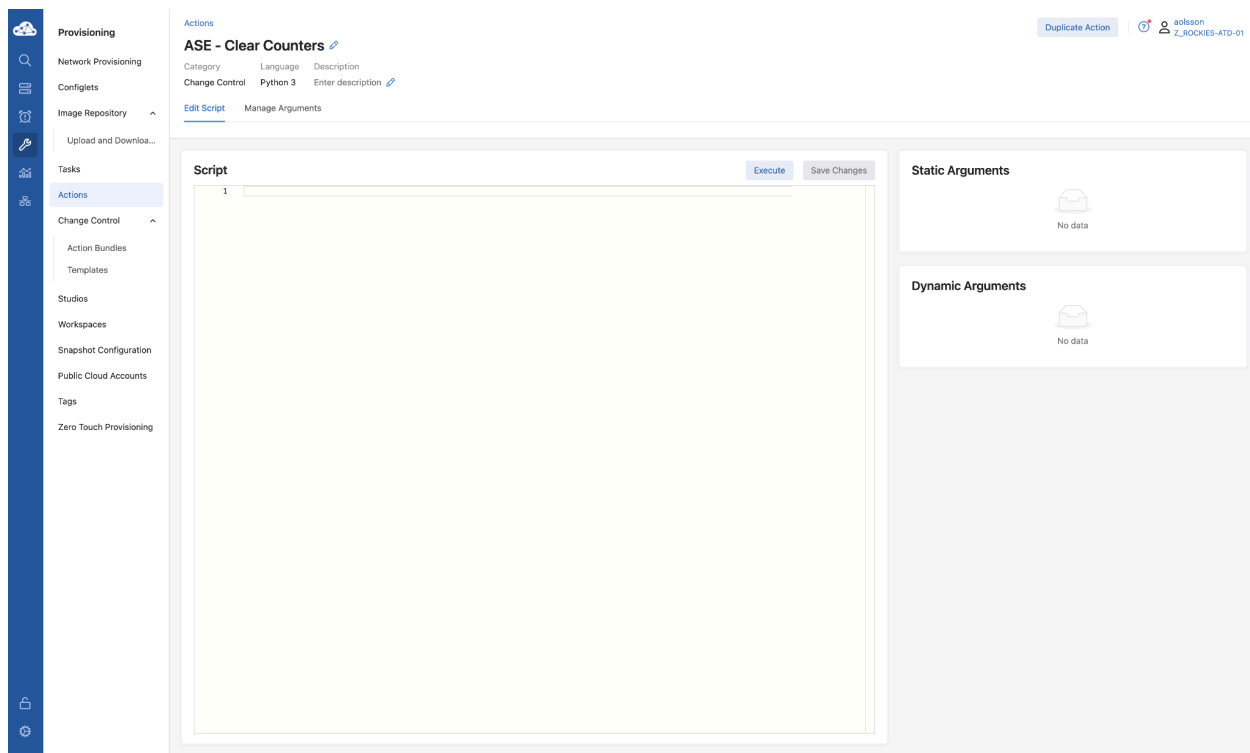
* Name
ASE - Clear Counters

Description

* Action Type
Change Control

Cancel Save

5. You'll then be redirected to the action configuration page for the new action.



6. We'll first want to create some arguments for the Device and Interface for the action to be run against. To do this, click the **"Manage Arguments"** button just below the action's name

The screenshot shows the Arista Provisioning web interface. On the left is a navigation sidebar with options like Provisioning, Network Provisioning, Configlets, Image Repository, Upload and Download, Tasks, Actions (selected), Change Control, Action Bundles, Templates, Studios, Workspaces, Snapshot Configuration, Public Cloud Accounts, Tags, and Zero Touch Provisioning. The main content area is titled 'Actions' and shows 'ASE - Clear Counters' with a 'Duplicate Action' button and a user profile 'adison_2_R00KES-ATD-01'. Below this, there's a 'Manage Arguments' section with a 'Save Changes' button. A table with columns 'Name', 'Type', 'Default', 'Description', 'Required', and 'Hidden' is shown, currently containing no data. At the bottom of the table are two buttons: '+ Static Argument' and '+ Dynamic Argument'.

7. Create a new argument by selecting the “+ **Dynamic Argument**” at the bottom of the “**Manage Arguments**” section.

NOTE: Keep an eye on the Arguments and Variables as they are case sensitive.

8. A new line for the new argument will appear in the “**Manage Arguments**” section. Enter the following into the new line:

```
Name : DeviceID
Type : Dynamic
Default : Leave Blank
Description : The ID of the device to run this script against
Required : Yes
Hidden : No
```

The screenshot shows the Arista Provisioning web interface. On the left is a navigation sidebar with options like Provisioning, Network Provisioning, Configlets, Image Repository, Upload and Download..., Tasks, Actions (selected), Change Control, Action Bundles, Templates, Studios, Workspaces, Snapshot Configuration, Public Cloud Accounts, Tags, and Zero Touch Provisioning. The main content area is titled 'Actions' and shows 'ASE - Clear Counters'. Below this, there are tabs for 'Edit Script' and 'Manage Arguments' (selected). The 'Manage Arguments' section includes a 'Save Changes' button and a table with columns: Name, Type, Default, Description, Required, and Hidden. The table contains one entry: 'DeviceID' with Type 'Dynamic', Description 'The ID of the device to run this script against', Required 'Yes', and Hidden 'No'. Below the table are links for 'Static Argument' and 'Dynamic Argument'.

Name	Type	Default	Description	Required	Hidden
DeviceID	Dynamic		The ID of the device to run this script against	Yes	No

9. Repeat steps 7 & 8 to create the interface argument using the following values:

```
Name : Interface
Type : Dynamic
Default : Leave Blank
Description : The interface number of the interface to clear counters on
Required : Yes
Hidden : No
```

The screenshot shows the Arista Provisioning interface. On the left is a sidebar with navigation options: Provisioning, Network Provisioning, Configlets, Image Repository, Upload and Download..., Tasks, Actions (selected), Change Control, Action Bundles, Templates, Studios, Workspaces, Snapshot Configuration, Public Cloud Accounts, Tags, and Zero Touch Provisioning. The main content area is titled 'Actions' and shows 'ASE - Clear Counters'. Below this, there are tabs for 'Edit Script' and 'Manage Arguments' (selected). The 'Manage Arguments' section includes a 'Save Changes' button and a table of arguments.

Name	Type	Default	Description	Required	Hidden
DeviceID	Dynamic		The ID of the device to run this script against	Yes	No
Interface	Dynamic		The interface number of the interface to clear counters on	Yes	No

Below the table are links for 'Static Argument' and 'Dynamic Argument'.

10. Click the blue **“Save Changes”** button on the right side of the screen.

11. Now that the Arguments are created, we can head to the Script section to begin writing the action. Click on the **“Edit Script”** button below the Action name.

The screenshot shows the Arista Provisioning interface with the 'Edit Script' tab selected. The main content area is titled 'Script' and includes an 'Execute' button and a 'Save Changes' button. The script editor is empty, with a line number '1' at the top. On the right side, there are sections for 'Static Arguments' (showing 'No data') and 'Dynamic Arguments' (showing 'DeviceID (Required)' and 'Interface (Required)' with input fields).

12. On line 1 of the script we'll want to identify who wrote the script. Go ahead and enter the following on line 1, replacing "your-name-here" with your name:

```
# Written by your-name-here
```

13. The next thing we need to do is create a variable in the script that references the "Interface" argument that was created. To do that, we skip a line and then add the following on lines 3 & 4.

```
# Create a variable to reflect the "Interface" dynamic argument
interface = ctx.action.args.get("Interface")
```

14. Now, we need to add two different lists of commands that will need to be run on the switch. The first list will show the interface's existing counters, and the second will clear the interface counters. Skip a line and then paste the following into lines 6 - 16:

```
# List of commands to show interface counters
showCountersCmds = [
    'enable',
    f'show interfaces ethernet{interface} counters'
]

# List of commands to clear interface counters
clearCountersCmds = [
    'enable',
    f'clear counters ethernet{interface}'
]
```

15. Validate that the script section matches the following screenshot.

The screenshot shows the Arista Provisioning web interface. On the left is a sidebar with navigation options: Provisioning, Network Provisioning, Configlets, Image Repository, Upload and Download..., Tasks, Actions (selected), Change Control, Action Bundles, Templates, Studios, Workspaces, Snapshot Configuration, Public Cloud Accounts, Tags, and Zero Touch Provisioning. The main area displays the 'ASE - Clear Counters' action. It has a category of 'Change Control', language of 'Python 3', and a description 'Enter description'. Below this is a 'Script' editor with the following code:

```

1 # Written by your-name-here
2
3 # Create a variable to reflect the "Interface" dynamic argument
4 interface = ctx.action.args.get("Interface")
5
6 # List of commands to show interface counters
7 showCountersCmds = [
8     'enable',
9     f'show interfaces ethernet {interface} counters'
10 ]
11
12 # List of commands to clear interface counters
13 clearCountersCmds = [
14     'enable',
15     f'clear counters ethernet {interface}'
16 ]

```

Buttons for 'Execute' and 'Save Changes' are visible. On the right, there are sections for 'Static Arguments' (showing 'No data') and 'Dynamic Arguments' (listing 'DeviceID (Required)' and 'Interface (Required)').

16. Let's continue editing the script by adding some logging stating that we're getting the values of the current interface counters. Skip another line and then add the following to lines 18 & 19:

```

# Create log entry
ctx.alog(f'Getting counters for Ethernet{interface}')

```

17. Now, we can proceed with running the commands listed in the "showCountersCmds" list on the switch. To do that, add another blank line and the following to lines 21 and 22.

```

# Run the list of commands to get counters
cmdResponse = ctx.runDeviceCmds(showCountersCmds)

```

18. The commands have now been run on the switch and the response has been stored in the variable "**cmdResponse**." Let's create a log entry that outputs the response. On lines 24 - 25 add the following:

```

# Create a log entry of the counters
ctx.alog(f'The current interface counters are : {cmdResponse[1]["response"]}')

```


NOTE: The cmdResponse value returned above is similar to the following:

```
[{'error': '', 'response': {}},
{'error': '', 'response': {
  'interfaces': {
    'Ethernet1': {
      'inBroadcastPkts': 1,
      'inDiscards': 0,
      'inMulticastPkts': 38,
      'inOctets': 7529,
      'inUcastPkts': 0,
      'lastUpdateTimestamp': 1717697811.8986592,
      'outBroadcastPkts': 836, 'outDiscards': 0,
      'outMulticastPkts': 37,
      'outOctets': 45227, 'outUcastPkts': 0}
    }
  }
}]
```

The cmdResponse list contains a dictionary for each command run on the device. Since we're interested in the "show interfaces ethernet counters" command response value, we need to specify "[1]" to indicate the second item in the list (since list values start at the 0th item in Python) and ["response"] for the response data.

19. Now that we have logged what the counters were, let's proceed with clearing the counters. Let's create a blank line, then create another log entry stating that we will clear the counters. On lines 27 - 28 enter the following:

```
# Create log entry before clearing the counters
ctx.alog(f'Clearing counters for Ethernet{interface}')
```

20. Let's actually clear the counters now. Let's skip another line and then enter the following on lines 30 - 31:

```
# Run the list of commands to clear counters
cmdResponse = ctx.runDeviceCmds(clearCountersCmds)
```

21. Now that the counters are cleared let's create another log entry stating that the counters have been successfully cleared. To do that, let's skip another line and add the following content to lines 33 - 34:

```
# Create log entry
ctx.alog(f'Counters for Ethernet{interface} Successfully Cleared')
```

22. For the final part of the script, let's repeat steps 16 - 18 to get the interface's current counter values and log them.
23. Validate that the script matches the screenshot below, then click the blue **“Save Changes”** button to save the action.

24. Now, we can proceed with testing our Action. To do so, click on the light blue **“Execute”** button.
25. An **“Execute Action”** dialog box will appear on the right side of the screen. Click **“Dynamic Argument Values”** to expand the arguments section.

The screenshot shows the Arista Provisioning console. On the left is a sidebar with navigation options like Provisioning, Network Provisioning, Configlets, Image Repository, Tasks, Actions, Change Control, Action Bundles, Templates, Studios, Workspaces, Snapshot Configuration, Public Cloud Accounts, Tags, and Zero Touch Provisioning. The main panel displays the 'ASE - Clear Counters' action script. The script is written in Python and includes comments and code for showing and clearing interface counters. On the right, the 'Execute Action' dialog is open, showing 'Dynamic Argument Values' for 'Interface' and 'DeviceID'. The 'Interface' field is empty, and the 'DeviceID' dropdown is set to '1'.

```
1 # Written by your-name-here
2
3 # Create a variable to reflect the "Interface" dynamic argument
4 interface = ctx.action.args.get("Interface")
5
6 # List of commands to show interface counters
7 showCountersCmds = [
8     'enable',
9     f'show interfaces ethernet {interface} counters'
10 ]
11
12 # List of commands to clear interface counters
13 clearCountersCmds = [
14     'enable',
15     f'clear counters ethernet {interface}'
16 ]
17
18 # Create log entry
19 ctx.log(f'Getting counters for Ethernet {interface}')
20
21 # Run the list of commands to get counters
22 cmdResponse = ctx.runDeviceCmds(showCountersCmds)
23
24 # Create a log entry of the counters
25 ctx.log(f'The current interface counters are : {cmdResponse[1]["response"]}')
26
27 # Create log entry prior to clearing the counters
28 ctx.log(f'Clearing counters for Ethernet {interface}')
29
30 # Run the list of commands to clear counters
31 cmdResponse = ctx.runDeviceCmds(clearCountersCmds)
32
33 # Create log entry
34 ctx.log(f'Counters for Ethernet {interface} Successfully Cleared')
35
36 # Create log entry
37 ctx.log(f'Getting counters for Ethernet {interface}')
38
39 # Run the list of commands to get counters
40 cmdResponse = ctx.runDeviceCmds(showCountersCmds)
41
42 # Create a log entry of the counters
43 ctx.log(f'The current interface counters are : {cmdResponse[1]["response"]}')
```

26. Let's use Ethernet1 for this test, so specify “1” for the “Interface.”

27. In the DeviceID dropdown, select “Leaf-1A”

28. Next, click the blue “Execute” button.

This screenshot shows the same Arista Provisioning console as the previous one, but with the 'Execute Action' dialog updated. The 'Interface' field now contains the value '1', and the 'DeviceID' dropdown menu has been expanded to show 'Leaf-1A' as the selected option. The blue 'Execute' button is now visible at the bottom right of the dialog.

29. As the action executes, it will provide the logging output in the pane below the “**Execute**” button. The oldest log entry will be at the bottom, so you should read the logs from the bottom up. You’ll be able to see that the current interface counters were displayed, they were then cleared, and the new counters were displayed.

The screenshot displays the Arista CloudVision interface. On the left is a navigation sidebar with options like Provisioning, Tasks, Actions, and Workspaces. The main panel shows the 'ASE - Clear Counters' action script, which is a Python script designed to clear interface counters on a specific device. The script includes comments and code for creating dynamic arguments, listing commands, and logging the results.

On the right, the 'Execute Action' pane is visible. It shows the 'Dynamic Argument Values' section with 'Interface' set to '1' and 'DeviceID' set to 'Leaf-1A'. Below this, there is a search bar for logs and a list of execution logs. The logs show the sequence of events: getting counters for Ethernet1, successfully clearing them, and then displaying the new (zeroed) counters.

```

1 # Written by your-name-here
2
3 # Create a variable to reflect the "Interface" dynamic argument
4 interface = ctx.action.args.get("Interface")
5
6 # List of commands to show interface counters
7 showCountersCmds = [
8     'enable',
9     f'show interfaces ethernet {interface} counters'
10 ]
11
12 # List of commands to clear interface counters
13 clearCountersCmds = [
14     'enable',
15     f'clear counters ethernet {interface}'
16 ]
17
18 # Create log entry
19 ctx.log(f'Getting counters for Ethernet{interface}')
20
21 # Run the list of commands to get counters
22 cmdResponse = ctx.runDeviceCmds(showCountersCmds)
23
24 # Create a log entry of the counters
25 ctx.log(f'The correct interface counters are : {cmdResponse[1]["response"]}')
26
27 # Create log entry prior to clearing the counters
28 ctx.log(f'Clearing counters for Ethernet{interface}')
29
30 # Run the list of commands to clear counters
31 cmdResponse = ctx.runDeviceCmds(clearCountersCmds)
32
33 # Create log entry
34 ctx.log(f'Counters for Ethernet{interface} Successfully Cleared')
35
36 # Create log entry
37 ctx.log(f'Getting counters for Ethernet{interface}')
38
39 # Run the list of commands to get counters
40 cmdResponse = ctx.runDeviceCmds(showCountersCmds)
41
42 # Create a log entry of the counters
43 ctx.log(f'The correct interface counters are : {cmdResponse[1]["response"]}')
  
```

The execution logs on the right show the following sequence:

- 25 seconds ago: The correct interface counters are : {'interfaces': {'Ethernet1': {'inBroadcastPkts': 0, 'inDiscards': 0, 'inMulticastPkts': 0, 'inOctets': 0, 'inUcastPkts': 0, 'lastUpdateTimestamp': 171770194.640033, 'outBroadcastPkts': 0, 'outDiscards': 0, 'outMulticastPkts': 0, 'outOctets': 2243680, 'outUcast... Expand
- 27 seconds ago: Counters for Ethernet1 Successfully Cleared
- 27 seconds ago: Clearing counters for Ethernet1
- 29 seconds ago: The correct interface counters are : {'interfaces': {'Ethernet1': {'inBroadcastPkts': 1, 'inDiscards': 0, 'inMulticastPkts': 7, 'inOctets': 2892, 'inUcastPkts': 16, 'lastUpdateTimestamp': 171770190.2773674, 'outBroadcastPkts': 0, 'outDiscards': 8, 'outMulticastPkts': 8, 'outOctets': 165612243, 'outUcast... Expand
- 31 seconds ago: Getting counters for Ethernet1

30. Now, your custom action is available to be used within a change control.

NOTE: Check out this document for more information on custom actions.

<https://www.arista.com/en/support/toi/cvp-2021-3-0/14901-ui-for-custom-action-scripts>

Arista also has a repo with some custom actions you might find useful. That repo can be accessed here: <https://github.com/aristanetworks/cloudvision-python-actions>