1 Table of Contents

1.1 Revision History

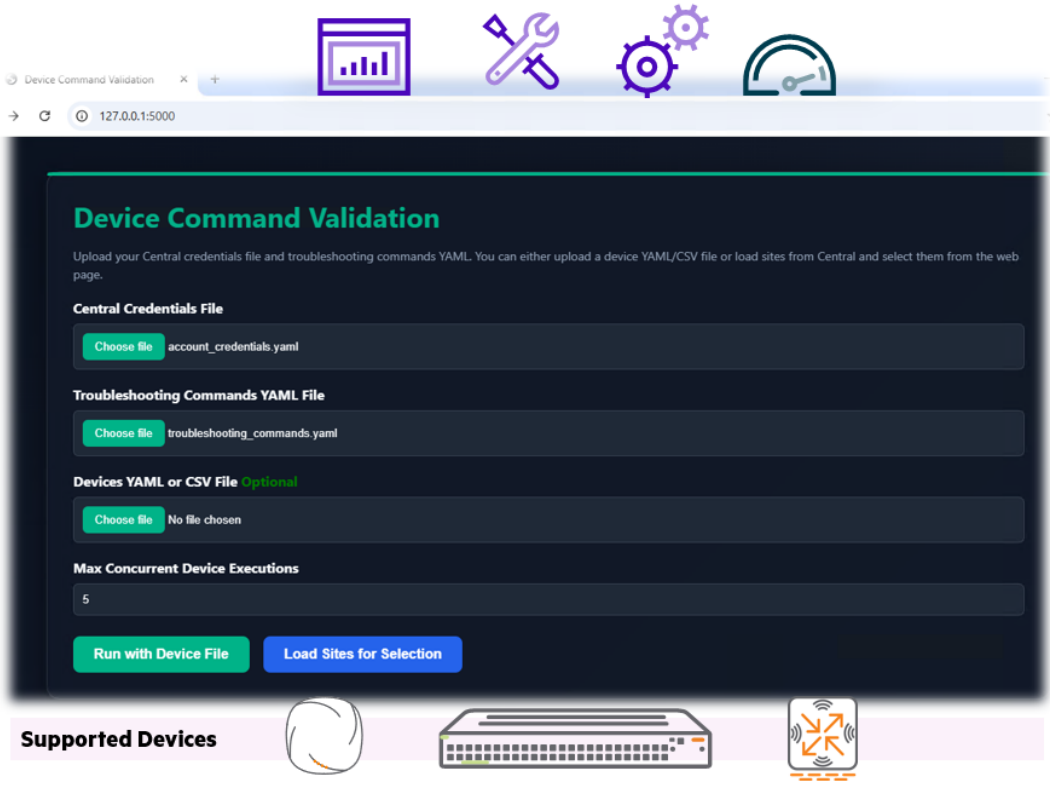
DATE	VERSION	EDITOR	CHANGES
16 Jul 2026	0.1	Ariya Parsamanesh	Initial creation

2 Validation Tool for New Central

The enhanced Cutover Validation Tool provides a guided workflow for validating Aruba Central-managed infrastructure during activities such as network cutovers, device migrations, firmware upgrades, site onboarding, operational troubleshooting, and validation of newly deployed environments.

While this information can be gathered directly from Aruba Central using the standard tools and UI, that process can take time during a deployment or cutover. In those situations, it is useful to have a quicker and more focused tool that can collect the required information, run validation checks, and generate a report that can be included in your as-built documentation.

The solution supports both command-line and web-based execution models. It uses Aruba Central APIs to retrieve device inventory, site information, device status, and firmware details, then runs the relevant validation and troubleshooting commands against supported device types.



This enhanced version is a fork of the Aruba Central Python Workflows project, with additional focus on cutover validation, firmware visibility, web-based execution, live progress tracking, and operational reporting.

https://github.com/ariyaparsa-dev/central-python-workflows/blob/feature/cutover-validation-enhancements/cutover-validation/README.md

The tool provides a web-based validation and troubleshooting workflow for Aruba Central environments. It allows operators to safely run validation commands against Access Points, CX switches, and AOS10 gateways while providing firmware visibility, real-time progress updates, and reports in multiple formats.

In this technote, I'll first cover how to configure API access for HPE Networking Aruba Central through HPE GreenLake. I'll then walk through how to configure and run the enhanced validation tool.

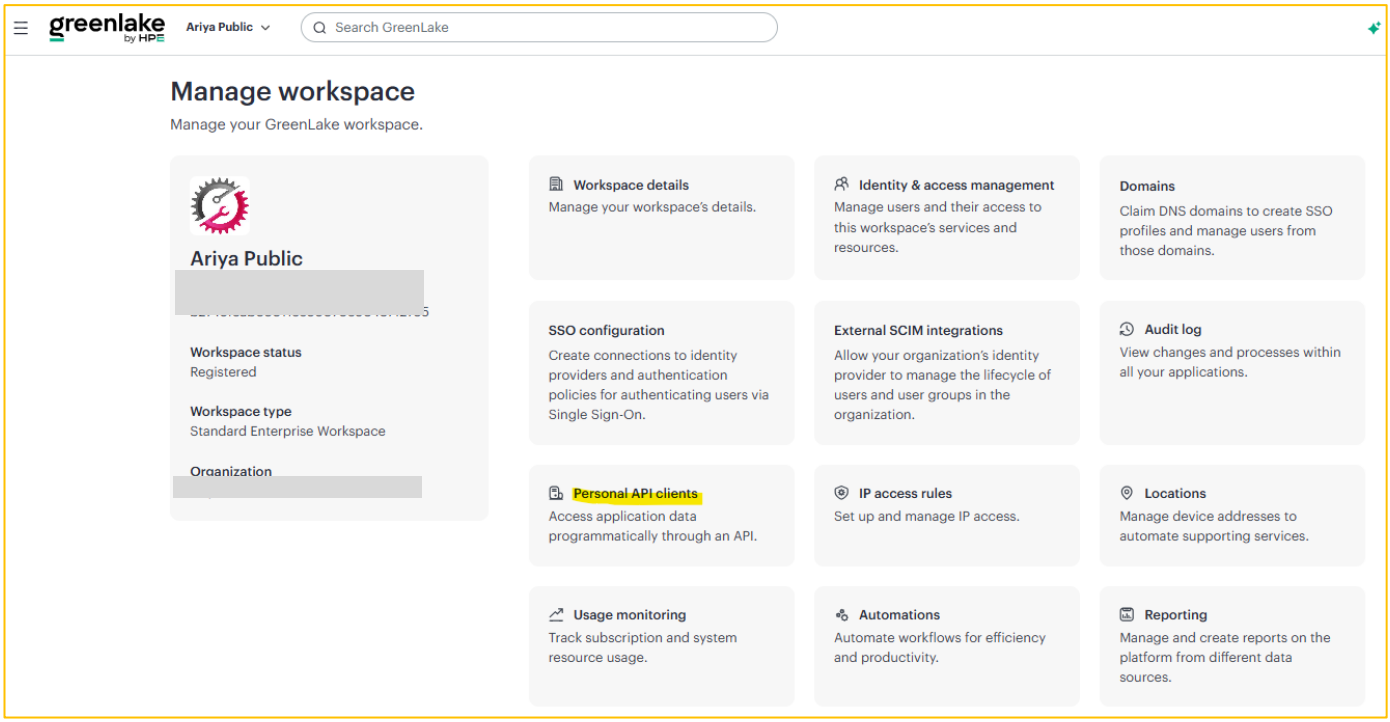
2.1 Things you need

- HPE Aruba Central account and subscriptions
- At least one site to have devices such as APs, CX switches and/or gateways.
- Python environment along with git bash. (I am using windows platform with Visual Studio code)

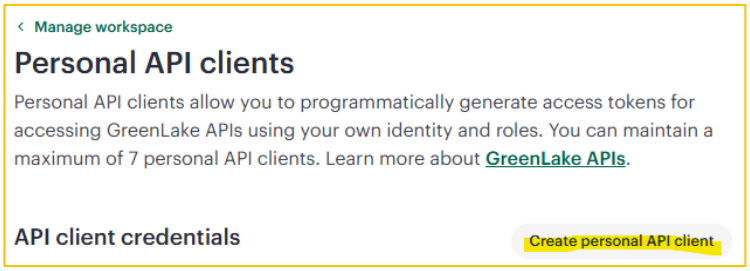
3 HPE GreenLake API

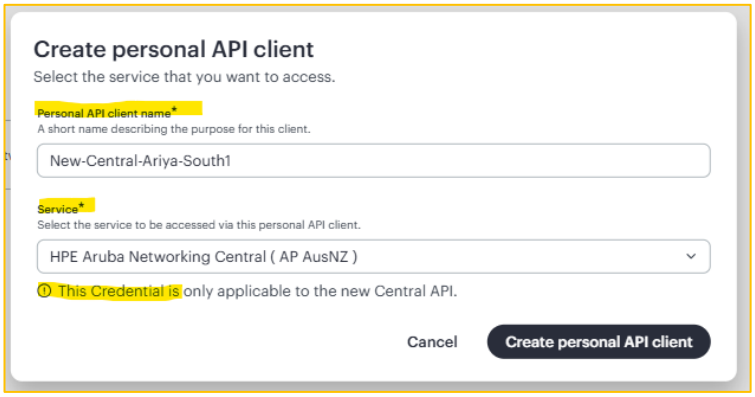
HPE GreenLake allows you to create a personal API client, which provides the client ID and client secret required to access GreenLake cloud services and related APIs. These credentials are used to generate an access token, which authenticates API communication between your application and GreenLake. In this guide, we'll use this access for the Aruba Central service.

First, configure API access for HPE Aruba Central so the validation tool can send API requests successfully. To do this, go to GreenLake >> Manage Workspace >> Personal API Clients.

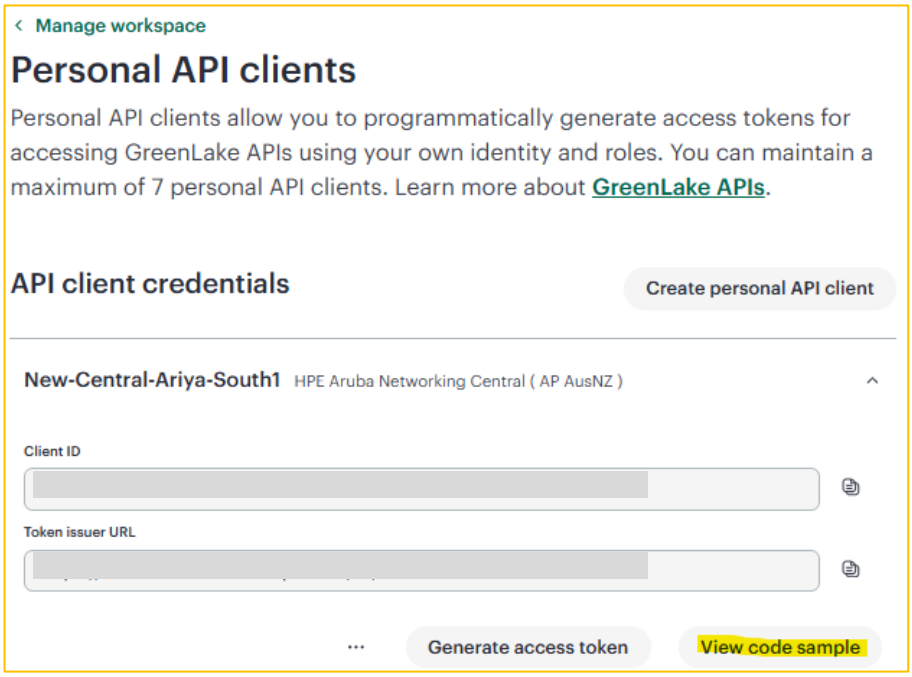


Here we need to create a new API client and select the services available for your account and in my case it is HPE Aruba Networking Central.



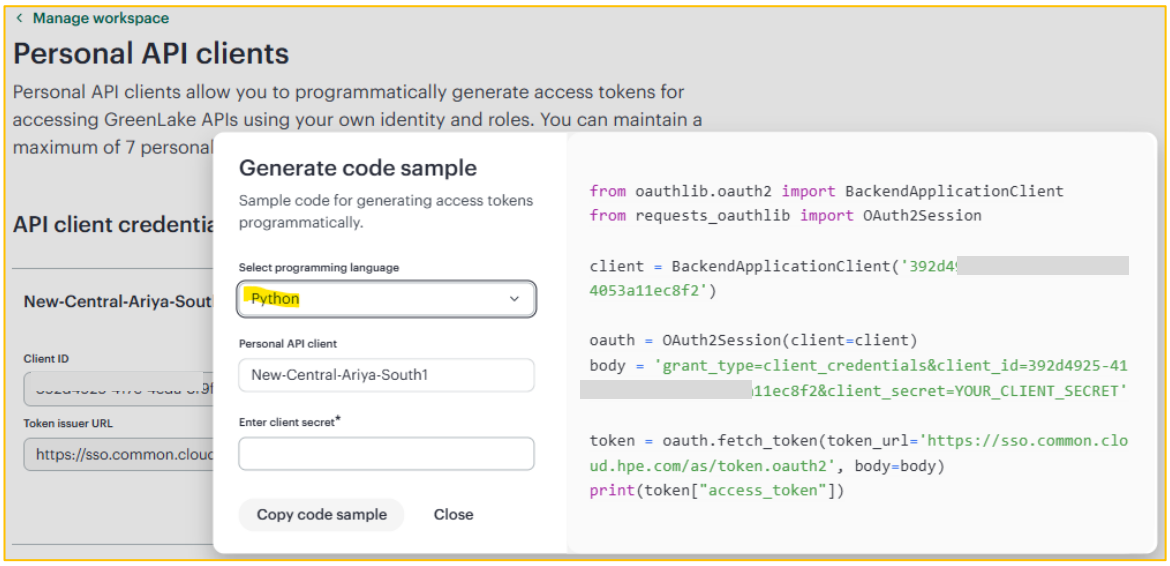


Once you have clicked on the create button, the personal API Client gets created and displayed as shown below.



Make a copy of the Client ID and the Client Secret which you'll need it for any API calls.

You can now click on the view code sample and choose python, and it provides the sample python code.



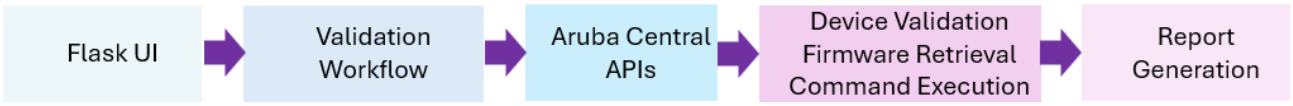
4 Enhanced Cutover Validation Tool

The tool provides a modern Flask-based web interface that guides users through the full validation workflow without requiring direct command-line interaction.

Key Capabilities

- Upload Aruba Central credentials
- Upload troubleshooting command files
- Discover Aruba Central sites automatically
- Select sites for validation
- Preview devices before execution
- Display firmware versions
- Launch validation jobs
- Monitor execution progress in real time
- View reports directly in the browser

The following section summarises the architecture and web UI technologies used by the tool.



Web UI Technologies

The web interface is built using the following components:

Technology	Purpose
Flask	Web framework
Jinja2	HTML template rendering
HTML5	UI structure
CSS3	Styling and responsive layout
JavaScript	Dynamic interactions
Fetch API	Live progress updates
Python	Validation workflow engine

4.1 Installation

The installation and configuration are easy. Once you have a working python environment. You need to perform three things.

1. Clone the repository:

```
git clone -b feature/cutover-validation-enhancements https://github.com/ariyaparsa-dev/central-python-workflows.git
```

Here is when you run it from a command prompt.

```
C:\Aruba\PythonScriptingPostman>git clone -b feature/cutover-validation-enhancements https://github.com/ariyaparsa-dev/central-python-workflows.git

Cloning into 'central-python-workflows'...
remote: Enumerating objects: 2262, done.
remote: Counting objects: 100% (552/552), done.
remote: Compressing objects: 100% (199/199), done.
remote: Total 2262 (delta 392), reused 353 (delta 353), pack-reused 1710 (from 1)
Receiving objects: 100% (2262/2262), 90.82 MiB | 9.20 MiB/s, done.
Resolving deltas: 100% (1113/1113), done.
Updating files: 100% (384/384), done.

C:\Aruba\PythonScriptingPostman>cd central-python-workflows
```

To ensure you have the correct branch type this command

```
C:\Aruba\PythonScriptingPostman\central-python-workflows>git branch
* feature/cutover-validation-enhancements

C:\Aruba\PythonScriptingPostman\central-python-workflows>
```

2. Change to the application directory:

```
cd central-python-workflows/cutover-validation
```

Here you can quickly check if the files have been downloaded with this command.

```
C:\Aruba\PythonScriptingPostman\central-python-workflows\cutover-validation>dir
 Volume in drive C is Windows

 Directory of C:\Aruba\PythonScriptingPostman\central-python-workflows\cutover-validation

14/07/2026  03:50 PM    <DIR>          .
14/07/2026  03:50 PM    <DIR>          ..
14/07/2026  03:50 PM            10,565 app.py
14/07/2026  03:50 PM            20,126 device_validation.py
14/07/2026  03:50 PM    <DIR>          images
14/07/2026  03:50 PM            12,344 main.py
14/07/2026  03:50 PM             7,621 README.md
14/07/2026  03:50 PM                41 requirements.txt
14/07/2026  03:50 PM               257 sample_account_credentials.yaml
14/07/2026  03:50 PM               351 sample_devices.yaml
14/07/2026  03:50 PM    <DIR>          sample_result_output
14/07/2026  03:50 PM               365 sample_troubleshooting_commands.yaml
14/07/2026  03:50 PM    <DIR>          Screenshots
14/07/2026  03:50 PM    <DIR>          templates
14/07/2026  03:50 PM    <DIR>          utils
14/07/2026  03:50 PM               187 valid_8_output.csv
               9 File(s)         51,857 bytes
               7 Dir(s)  77,373,718,528 bytes free

C:\Aruba\PythonScriptingPostman\central-python-workflows\cutover-validation>
```

3. Install dependencies:

```
pip install -r requirements.txt
```

```
C:\Aruba\PythonScriptingPostman\central-python-workflows\cutover-validation>pip
install -r requirements.txt

Collecting pycentral==2.0a22 (from -r requirements.txt (line 1))
  Downloading pycentral-2.0a22-py3-none-any.whl.metadata (7.5 kB)
Collecting tabulate==0.9.0 (from -r requirements.txt (line 2))
  Downloading tabulate-0.9.0-py3-none-any.whl.metadata (34 kB)
Collecting flask (from -r requirements.txt (line 3))
  Downloading flask-3.1.3-py3-none-any.whl.metadata (3.2 kB)
Requirement already satisfied: PyYAML==6.0.3 in
C:\Users\parsaman\AppData\Local\Python\pythoncore-3.14-64\Lib\site-packages (from
pycentral==2.0a22->-r requirements.txt (line 1)) (6.0.3)
Requirement already satisfied: oauthlib==3.2.2 in
C:\Users\parsaman\AppData\Local\Python\pythoncore-3.14-64\Lib\site-packages (from
pycentral==2.0a22->-r requirements.txt (line 1)) (3.2.2)
Requirement already satisfied: requests_oauthlib==2.0.0 in

<Lines omitted>

Downloading requests-2.33.0-py3-none-any.whl (65 kB)
Downloading flask-3.1.3-py3-none-any.whl (103 kB)
Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)
Downloading click-8.4.2-py3-none-any.whl (119 kB)
Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Downloading jinja2-3.1.6-py3-none-any.whl (134 kB)
Downloading markupsafe-3.0.3-cp314-cp314-win_amd64.whl (15 kB)
Downloading werkzeug-3.1.8-py3-none-any.whl (226 kB)
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
```

```
Installing collected packages: tabulate, requests, pycountry, markupsafe,
itsdangerous, colorama, blinker, werkzeug, jinja2, click, flask, pycentral
  Attempting uninstall: requests
    Found existing installation: requests 2.32.5
    Uninstalling requests-2.32.5:
      Successfully uninstalled requests-2.32.5
  Attempting uninstall: pycentral
    Found existing installation: pycentral 2.0a17
    Uninstalling pycentral-2.0a17:
      Successfully uninstalled pycentral-2.0a17
Successfully installed blinker-1.9.0 click-8.4.2 colorama-0.4.6 flask-3.1.3
itsdangerous-2.2.0 jinja2-3.1.6 markupsafe-3.0.3 pycentral-2.0a22 pycountry-26.2.16
requests-2.33.0 tabulate-0.9.0 werkzeug-3.1.8

C:\Aruba\PythonScriptingPostman\central-python-workflows\cutover-validation>
```

At this stage we need to move to the configuration section.

4.2 Configuration

There are three yaml based configuration files.

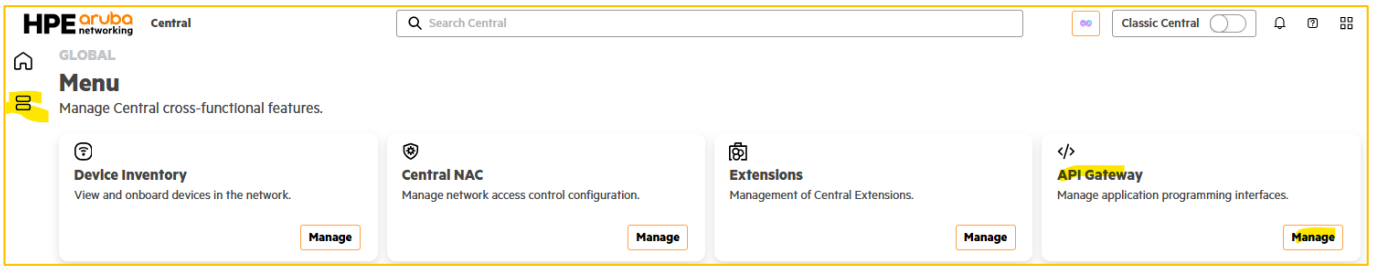
1. sample_account_credentials.yaml
2. sample_troubleshootin_commands.yaml
3. sample_devices.yaml

At minimum you need the first two that provide the API credentials to send the CLI commands to your Aruba central account.

Here is the sample_account_credentials.yaml

```
new_central:
  base_url: <use the correct base url for your central account>
  client_id: <add your client id here>
  client_secret: <add your client secret here>
  access_token: <add your access token here>
```

You can find HPE Networking Aruba Central account base URL by navigating to API gateway section.



Next, here is the sample_troubleshootin_commands.yaml

```
# Troubleshooting Commands YAML
# These commands will be executed on all devices in the device list
```

```
#
ap:
  - show ap debug cloud-server
  - show ap association
  - show version

cx:
  - show version
  - show vlan
  - show interface brief

gateway:
  - show version
  - show datapath session
```

4.3 Testing

The Cutover Validation Tool uses Flask, a lightweight Python web framework, to deliver a simple browser-based interface for configuring, running, and reviewing validation results.

Flask acts as the application backend and is responsible for:

- Serving the web interface
- Processing user requests
- Managing workflow execution
- Reading and validating configuration files
- Triggering cutover validation tasks
- Collecting validation results
- Generating reports for display and download

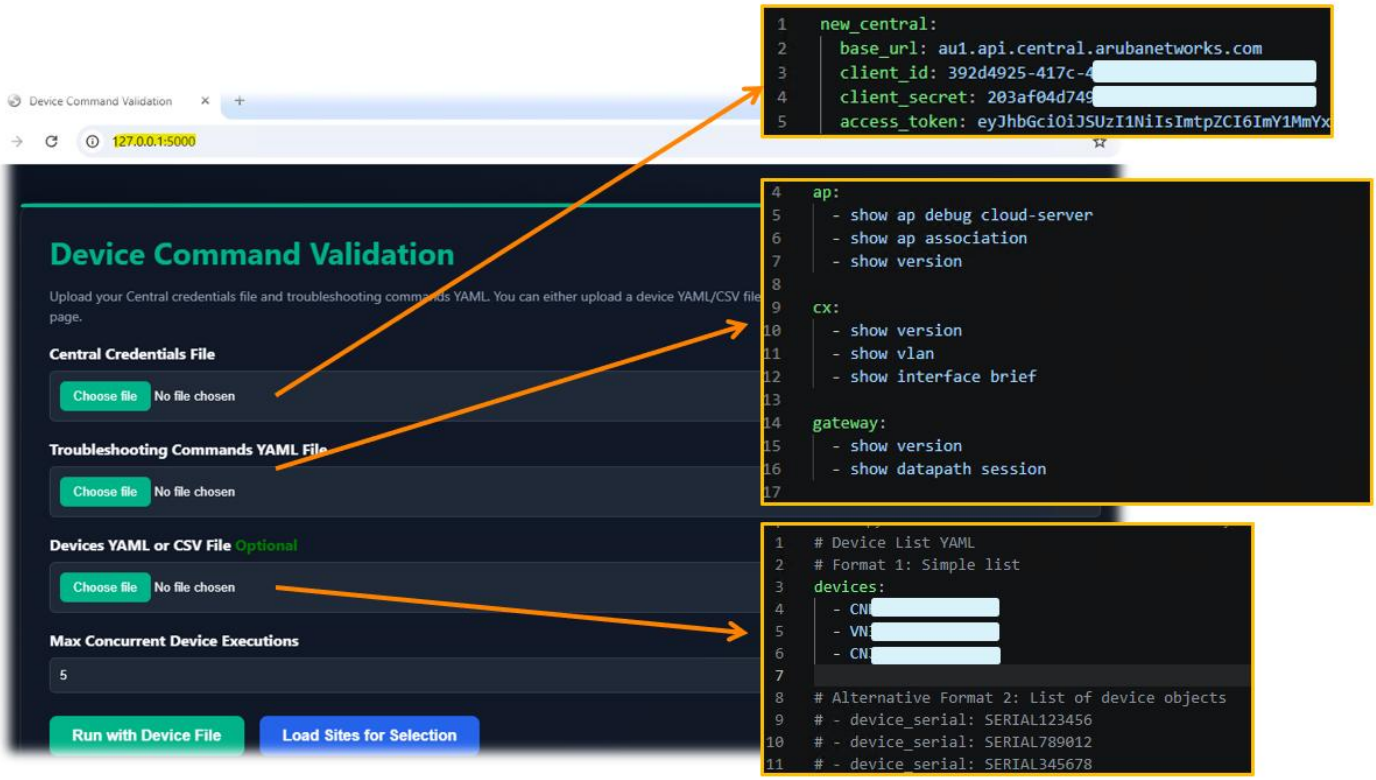
By using Flask, the tool can run as a lightweight local web application without requiring additional infrastructure. Engineers and operations teams can launch the tool locally and access it through a browser, making cutover validation easier to perform without working directly with Python scripts or command-line commands.

To start the web interface, run the command shown below. This launches the Python-based web front end for the validation tool.

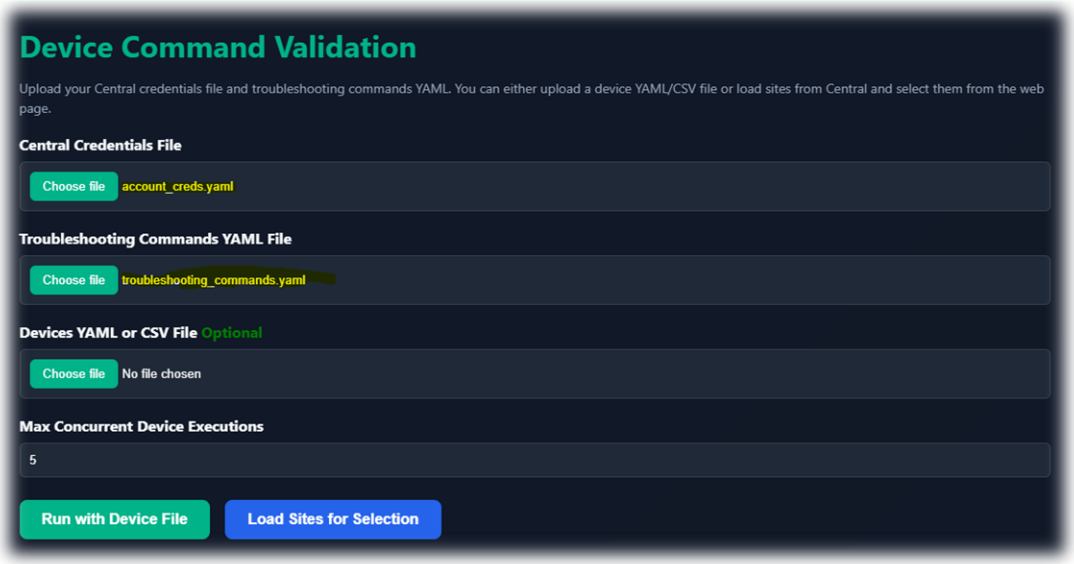
```
C:\Aruba\PythonScriptingPostman\central-python-workflows\cutover-validation>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 100-659-507
```

As indicated above you need to browse to 127.0.0.1:5000 to open the validation interface.

When the page loads successfully, you can select the required configuration files. The devices.yaml file is optional; if it is not provided, the tool can discover available sites and devices directly from Aruba Central.



As shown above, you can select your configuration files. Selecting “devices.yaml” is optional.

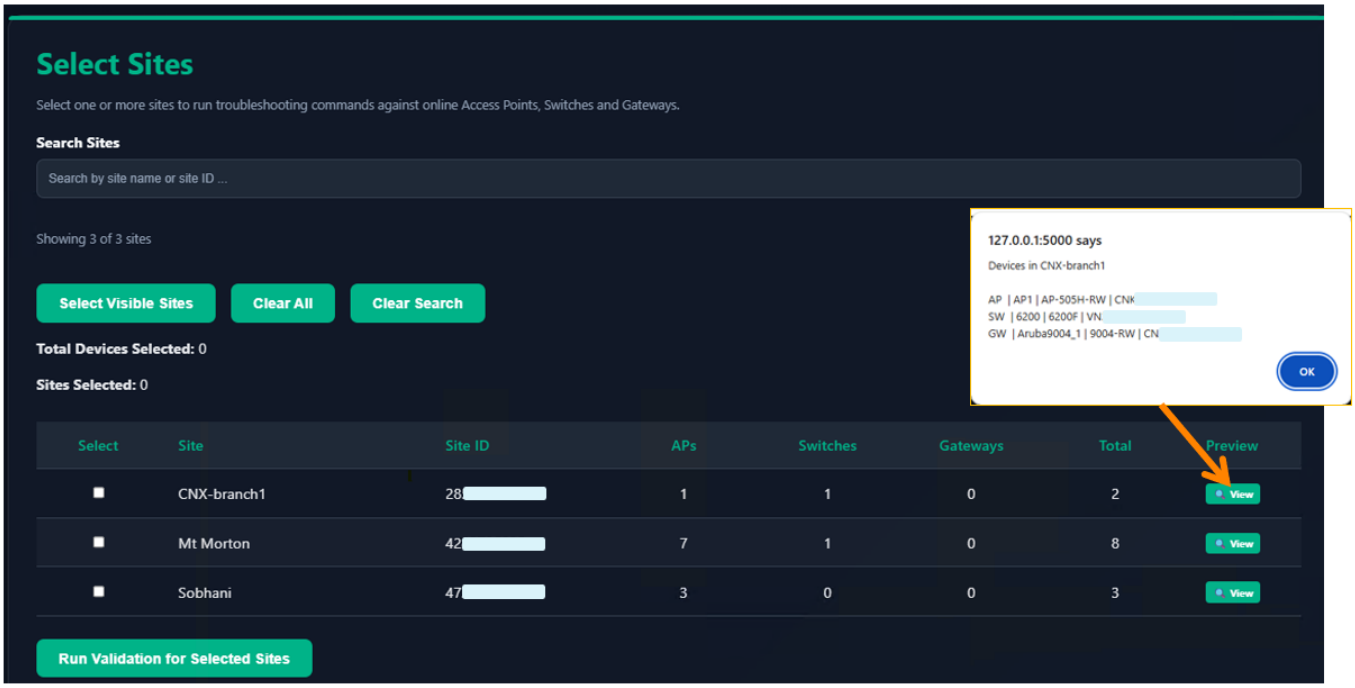


In this example, I have only selected the credentials and troubleshooting command YAML files. When you click Load sites for selection, the tool connects to Aruba Central, discovers the available sites and devices, retrieves firmware details, and then prepares the validation workflow.

The full workflow is shown below.

1. Connect to Aruba Central
2. Discover available sites and devices
3. Retrieve firmware information
4. Select devices or sites
5. Identify device types
6. Select device-specific commands
7. Validate troubleshooting commands
8. Execute supported commands
9. Monitor execution progress
10. Generate reports

In my case, I have just deployed the CNX-branch1 site, so I'll clear the current site selection and choose only that site. I can also click Preview to confirm which devices were discovered before I run the validation.



Once I'm happy that the deployed devices are listed correctly, I can click Run Validation to start the workflow.

At this point, the tool starts the validation workflow and shows live progress, giving me clear visibility into what is happening as each device is processed.

The progress view includes details such as:

- Completion percentage
- Successful devices
- Failed devices
- Last completed device
- Execution status

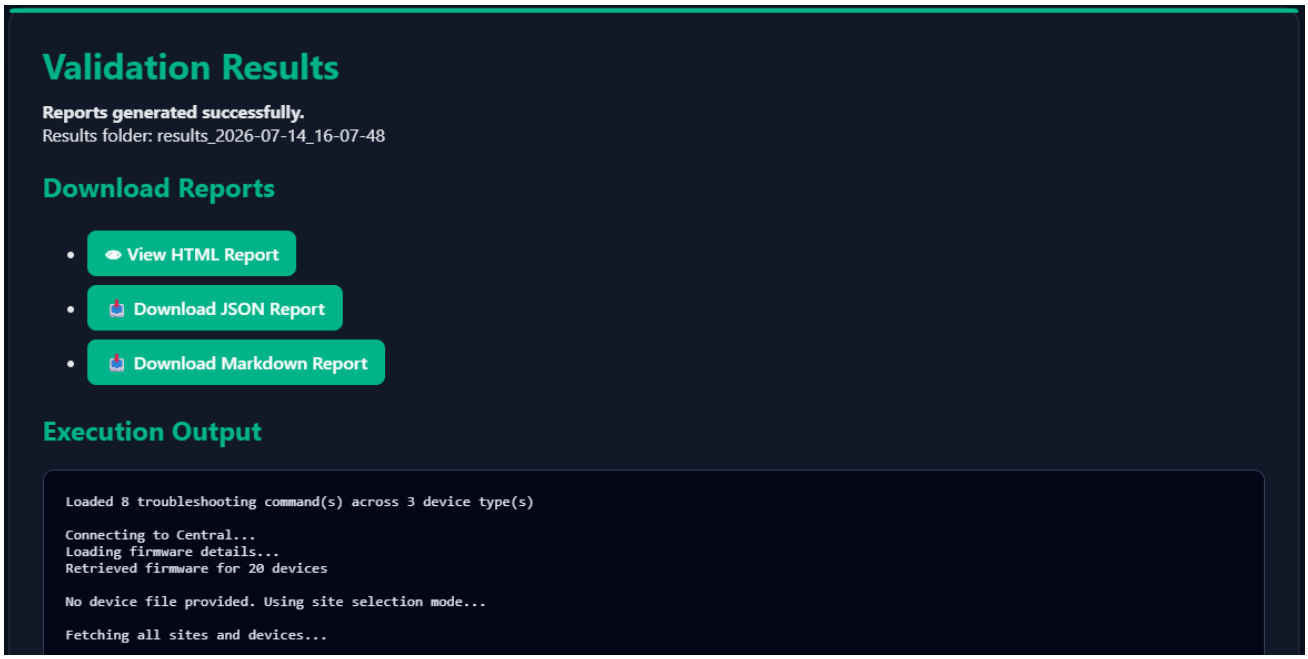
You can see an example of this progress view below.

It is also worth noting that the tool includes several safeguards before and during execution:

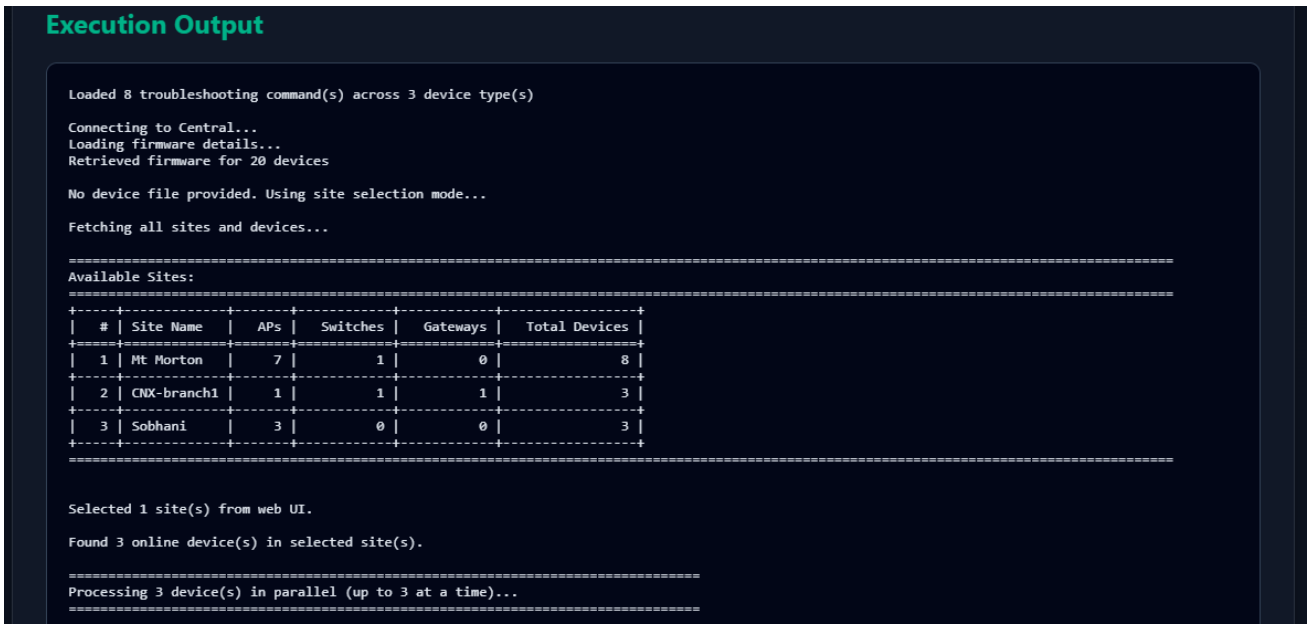
- Offline devices are skipped

- Devices without site assignment are skipped
- Invalid commands are identified and excluded
- Device status verification before execution
- Execution confirmation prompts

Once I click View Results, the tool opens a results page that includes the report in HTML, JSON, and Markdown formats, along with the execution output.

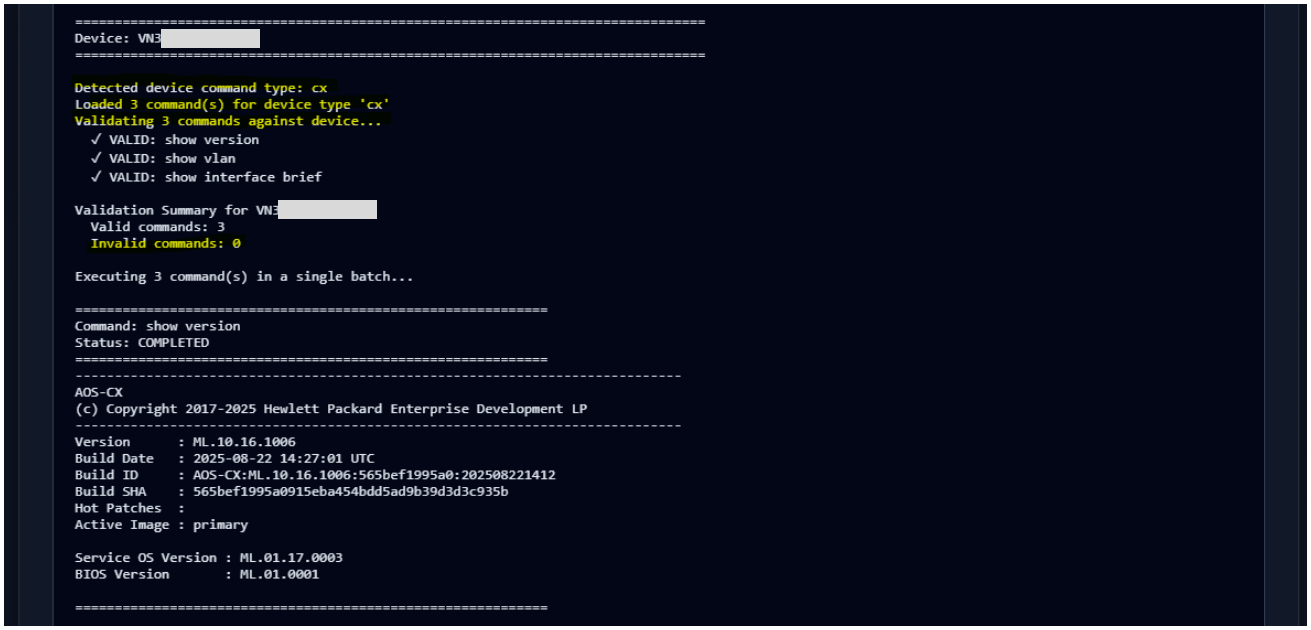


Before we jump into the HTML report, let's first take a quick look at the execution output.



The output shows that three devices were selected, and the tool will run the relevant troubleshooting commands based on each device type.

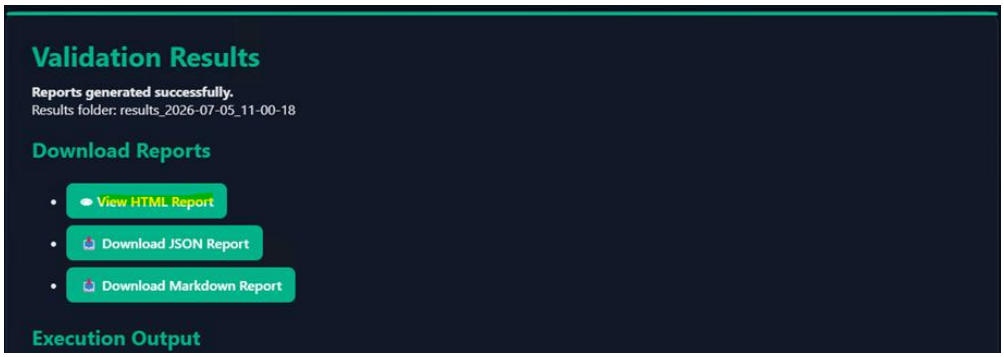
As I scroll down, I can see the command output grouped by device type, such as APs, switches, and gateways.



In the screenshot above, the tool validates each command against the relevant device type and highlights any commands that are not valid. At the end of the page, it confirms that the report was generated successfully and shows the directory where the report files are stored.

4.4 Validation Report

Now back to the top of that page you have a button to “view HTML Report”.



So, clicking on that will open another tab showing the output as shown below.

File C:/scripts/central-python-workflows/cutover-validation/results_2026-07-07_22-09-34/2026-07-07_22-09-34.html

Cutover Validation Report

Generated: 2026-07-07 22:09:34
Total Devices: 3
Total Commands: 8

Device Overview

#	Serial	Device Name	Model	IP Address	Firmware	Site	Status	Commands
1	CN	Aruba9004_1	9004-RW	192.168.1.241	10.7.2.5_95489	CNX-branch1	ONLINE	2
2	CN	AP1	AP-505H-RW	10.10.10.34	10.7.2.5_95489	CNX-branch1	ONLINE	3
3	VN	6200	6200F	10.10.10.11	ML.10.16.1006	CNX-branch1	ONLINE	3

Device CN

Serial Number: CNJFKL801Q | **Name:** Aruba9004_1 | **Model:** 9004-RW | **IP:** 192.168.1.241 | **Commands Executed:** 2

Command 1: show version

COMPLETED

```
## Command: show version - Tue Jul 14 16:07:38 2026
================================================
HPE Aruba Networking Wireless Operating System.
AOS-10 (MODEL: Aruba9004), Version 10.7.2.5 SSR
Website: http://www.arubanetworks.com
(c) Copyright 2026 Hewlett Packard Enterprise Development LP.
Compiled on 2026-04-03 at 19:36:59 UTC (build 95489) by jenkins

BIOS Version: INSYDE Corp., 9004.0026
Built: 06/05/2025

Switch uptime is 22 minutes 25 seconds
Reboot Cause: AC Power Cycle (Intent:cause: 86:50)
Supervisor Card
Processor(s):
Total CPUs : 4,    Sockets : 1,    Cores Per CPU :  4
       Socket 0: Intel(R) Atom(TM) CPU C3508 @ 1.60GHz
7541M bytes of memory
15028M bytes of Supervisor Card system flash.
```

Command 2: show datapath session

COMPLETED

```
Datapath Session Table Entries
```