1 Table of Contents

1 Table of Contents 1
1.1 Revision History 1
2 ClearPass Policy Visualiser 2
2.1 Things you need 3
3 ClearPass Configuration 4
3.1 NAD Configuration 4
3.2 API Operation 4
3.3 API Client Configuration 4
3.4 Test API Transaction 6
3.5 PostgreSQL Endpoint Profiling Access 8
4 ClearPass Policy Visualiser Application 9
4.1 Installation 9
4.2 Configuration 12
4.3 ClearPass Validation 17
4.4 ClearPass Policy Visualiser Configuration 19
4.5 ClearPass Policy Visualiser Console 21
4.6 Managing Multiple ClearPass Environments 23
4.7 Connection Diagnostics 24
5 ClearPass Policy Visualiser Operation 25
5.1 Dashboard 25
5.2 Visualisation and Interactive Dependency Graph 26
5.3 Advanced Search 28
5.4 Impact Analysis 30
5.5 Endpoint Matching 33
5.6 Unused Objects 35
6 Appendix A – Large Scale ClearPass Deployment Validation 37
6.1 Test Summary 37
6.2 Docker Console Log 37
7 Appendix B – CLI based ClearPass Policy Visualiser Configuration recovery 39

1.1 Revision History

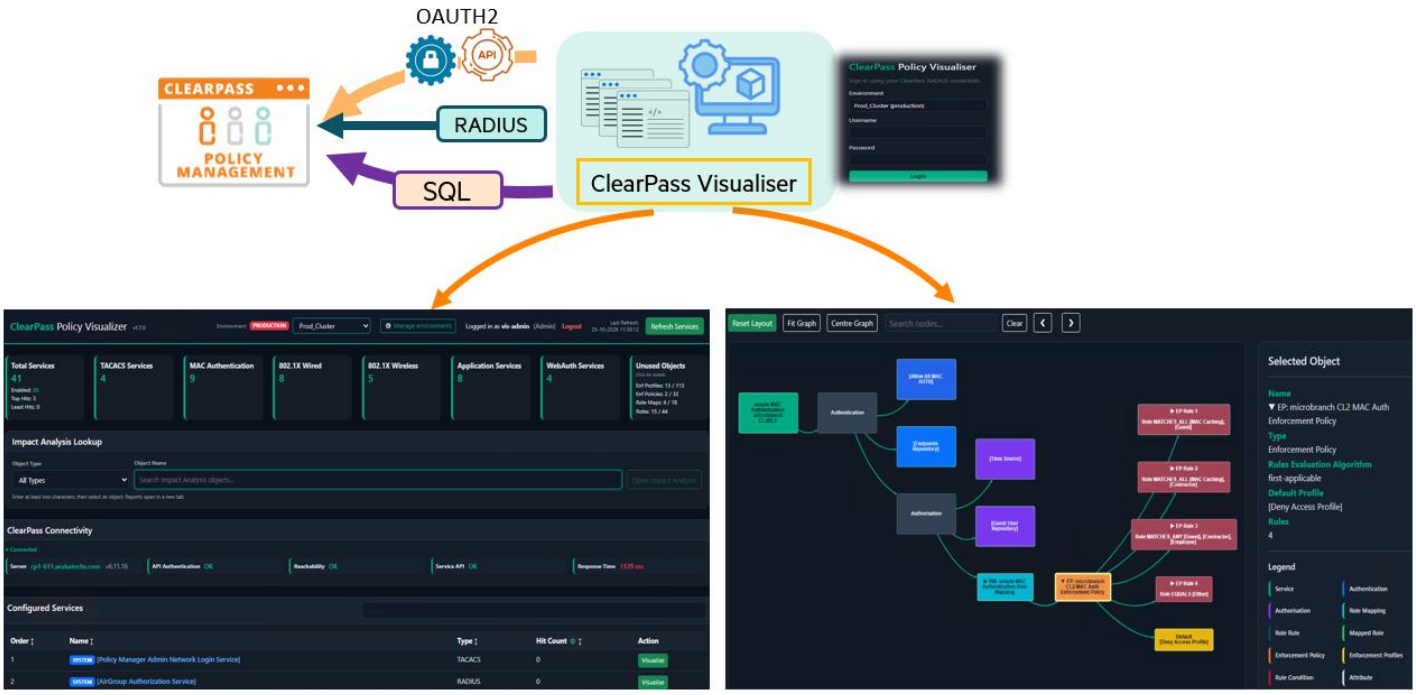
DATE	VERSION	EDITOR	CHANGES
09 Aug 2026	0.1	Ariya Parsamanesh	Initial creation
22 Aug 2026	0.2	Ariya Parsamanesh	Added the Visualise API assisted configuration.
12 Sep 2026	0.3	Ariya Parsamanesh	Added v1.6.0 enhancements, including Docker-based deployment and direct ClearPass PostgreSQL access for endpoint profiling.
29 Sep 2026	0.4	Ariya Parsamanesh	Added admin operator profile permissions and Appendix A large scale ClearPass validation.
29 Oct 2026	0.5	Ariya Parsamanesh	Added v1.7.0 features: multiple ClearPass environments, the Environments page, editing of the initial env, connection diagnostics and RADIUS check.

2 ClearPass Policy Visualiser

ClearPass Policy Visualiser (CPV) is a dependency analysis and policy impact exploration platform for HPE Aruba Networking ClearPass Policy Manager. The platform provides an interactive view of the relationships between Services, Role Mapping Policies, Enforcement Policies, Enforcement Profiles, Roles and Enforcement Attributes. It helps administrators trace dependencies, inspect policy logic, identify unused objects and understand how proposed changes may affect authentication, authorisation and enforcement workflows.

Version 1.7.0 lets one CPV instance manage up to three ClearPass environments, such as Production, Test and Dev, from a new Environments page. It also adds plain-language connection diagnostics and a RADIUS check. These build on the Docker-based deployment and direct PostgreSQL endpoint profiling introduced in v1.6.0, and on the read-only Impact Analysis, dashboard lookup, endpoint matching and unused-object analysis.

By consolidating policy relationships and dependencies into one interface, ClearPass Policy Visualiser supports troubleshooting, change planning, configuration review, cleanup and operational documentation.



https://github.com/ariyaparsa-dev/clearpass-policy-visualiser

Key Benefits

- Visualise complete ClearPass workflows and policy relationships in one interactive interface.
- Assess dependencies and change impact across Services, policies, profiles, Roles and default outcomes.
- Accelerate troubleshooting and validation through interactive graphs, condition analysis and endpoint matching.
- Improve configuration hygiene by identifying unused and unreferenced ClearPass objects.
- Simplify reviews and documentation with searchable policy data and direct navigation between related objects.
- Simplify deployment and administration through Docker-based installation, guided setup, validation and optional API-assisted ClearPass configuration.
- Manage several ClearPass environments, such as Production, Test and Dev, from one CPV instance, each with its own connection settings and sign-in.

In this technote, I'll first cover how to configure ClearPass API access and then provide step-by-step guidance for installing CPV using Docker, completing the initial configuration and exploring its main operational capabilities.

For the latest version of CPV always refer to the GitHub repository.

2.1 Things you need

Before we get started, you'll need:

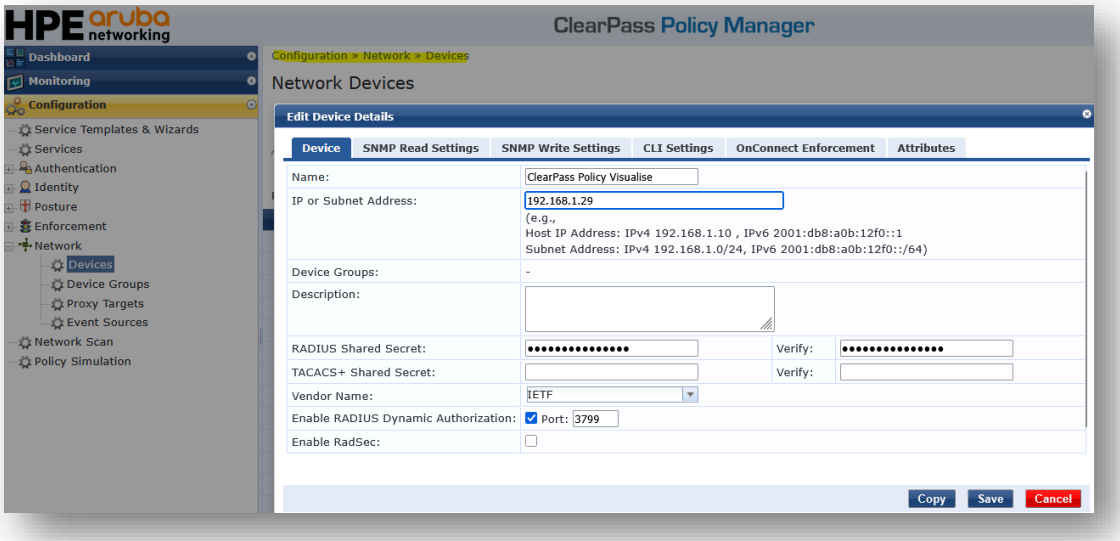
- HPE Aruba Networking ClearPass Policy Manager
- A ClearPass REST API client with the required permissions
- Access to the ClearPass PostgreSQL database using the appexternal account for endpoint profiling
- Docker with Docker Compose support on the system that will run CPV
- Git program to clone the ClearPass Policy Visualiser repository
- Network access from the CPV host to each ClearPass on TCP 443 (REST API), UDP 1812 (RADIUS), TCP 5432 (PostgreSQL) and TCP 5433 (Insight)

I'm using a Windows laptop for this technote, but the Docker-based deployment removes the need to install and maintain a separate Python runtime and application dependencies on the host.

3 ClearPass Configuration

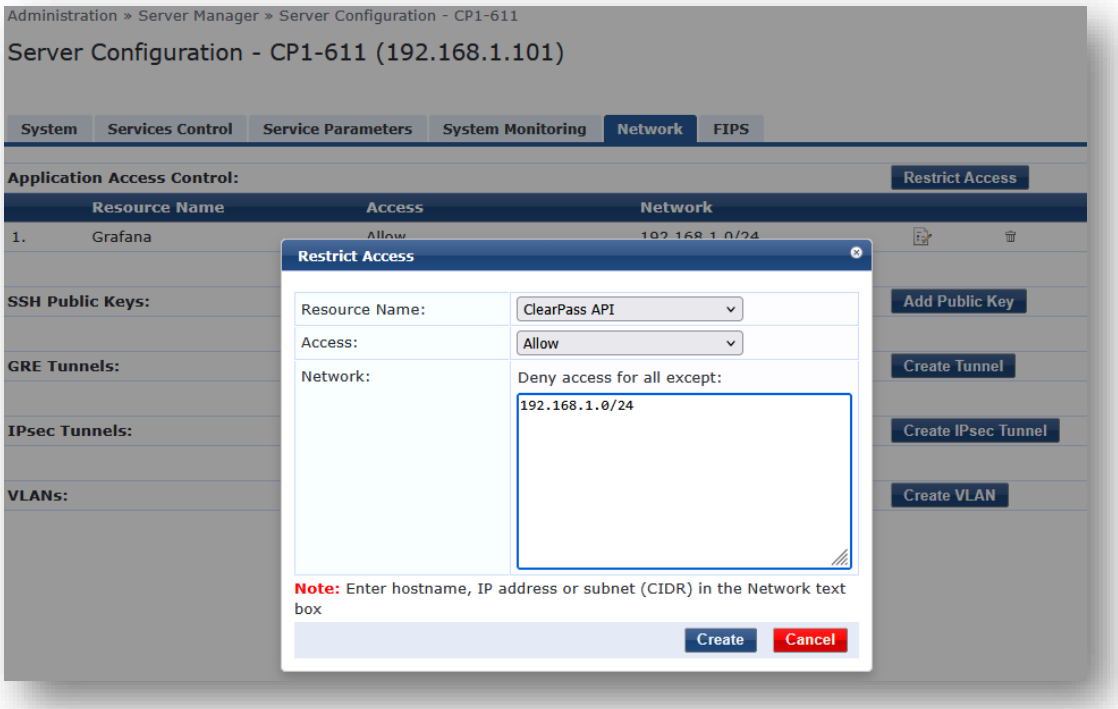
3.1 NAD Configuration

I'll start with adding CPV as a NAD in ClearPass so that it can perform authentication. Note the RADIUS shared secret as we need it during the initial setup of ClearPass Policy Visualiser.



3.2 API Operation

The next step is adding some restriction around API calls and where should it come from. As shown only the following subnet can initiate an API call. My CPV will be on this subnet.



3.3 API Client Configuration

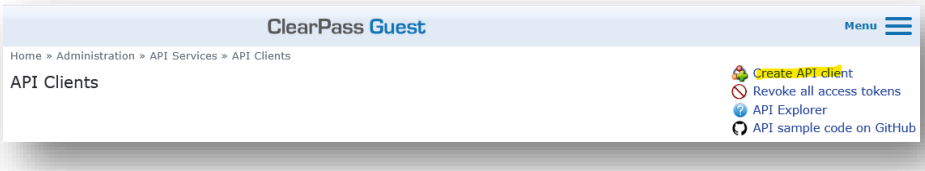
OAuth is an authorisation framework allowing users to grant websites or applications limited access to their resources without exposing their credentials. Grant types are the specific methods an application uses to request that access.

The following table shows the two different methods.

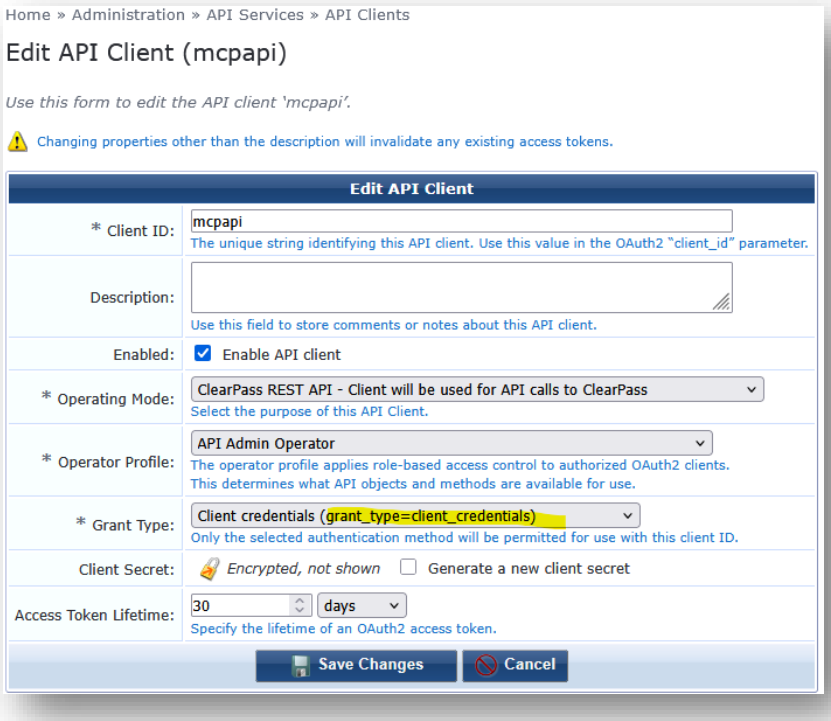
Feature	grant_type = password	grant_type = client_credentials

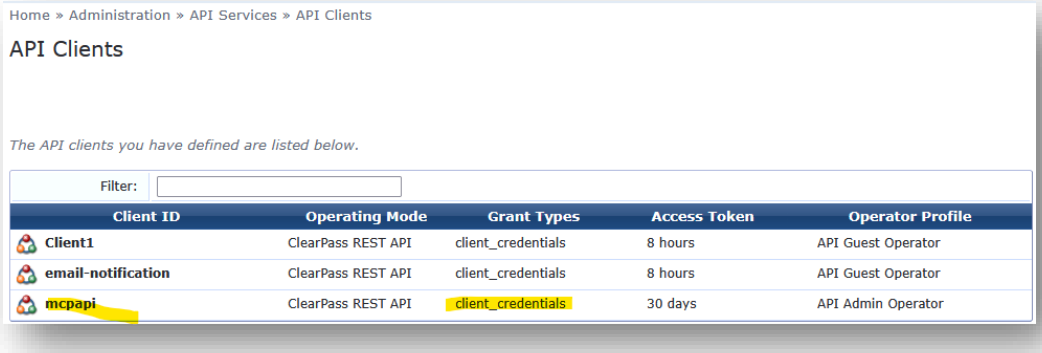
Who initiates it	The end-user (via a client application)	The client application/machine itself
Credentials used	User's username + password + client ID/secret	Client ID + client secret only
Use Case	First-party highly trusted apps requiring direct user logins (e.g., legacy native apps)	Backend-to-backend API calls, cron jobs, background microservices

Here, we'll use the grant type to client_credentials when you create your new API client.



Here you choose the operating Profile being API Admin Operator and then choose grant_type of Client credentials. This will display your client secret. Make sure you copy that as you'll need it when you make API calls. This is not shown here.

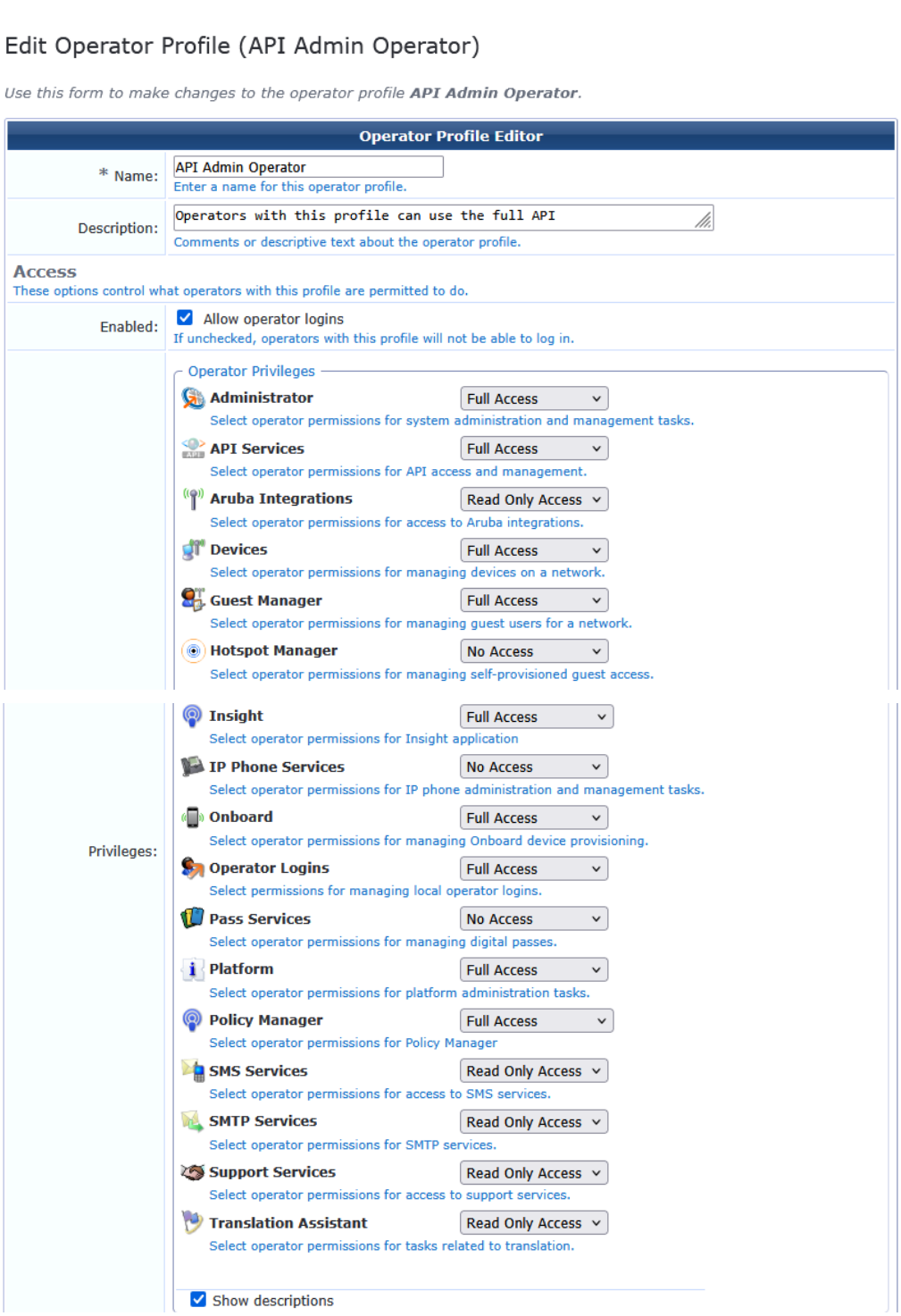




This approach does not need a dedicated OAuth service in the policy manager.

For API Admin Operator's the access privileges navigate to Home » Administration » Operator Logins » Profiles

This is the operator profile that was used.



3.4 Test API Transaction

I am assuming that the python environment is already setup and pyclearpass SDK is already in place. Here is a simple Python script that uses the pyclearpass SDK and a config.yaml file to perform an API call to ClearPass.

For more info on pyclearpass refer to https://developer.arubanetworks.com/cppm/docs/getting-started-with-pyclearpass

The yaml file is as shown below, and it has the details for the API call.

```
clearpass:
  server: https://192.168.1.101:443/api
  grant_type: client_credentials
  client_secret:
  client_id: mcpapi
  verify_ssl: false
```

And this is my simple python script called clearpass_roles.py that basically display various configured ClearPass roles.

```
from pyclearpass import *
```

```
import yaml

# Load configuration
with open("config.yaml", "r") as f:
    config = yaml.safe_load(f)

cp = config["clearpass"]

login = ClearPassAPILogin(
    server=cp["server"],
    granttype=cp["grant_type"],
    clientsecret=cp["client_secret"],
    clientid=cp["client_id"],
    verify_ssl=cp["verify_ssl"]
)

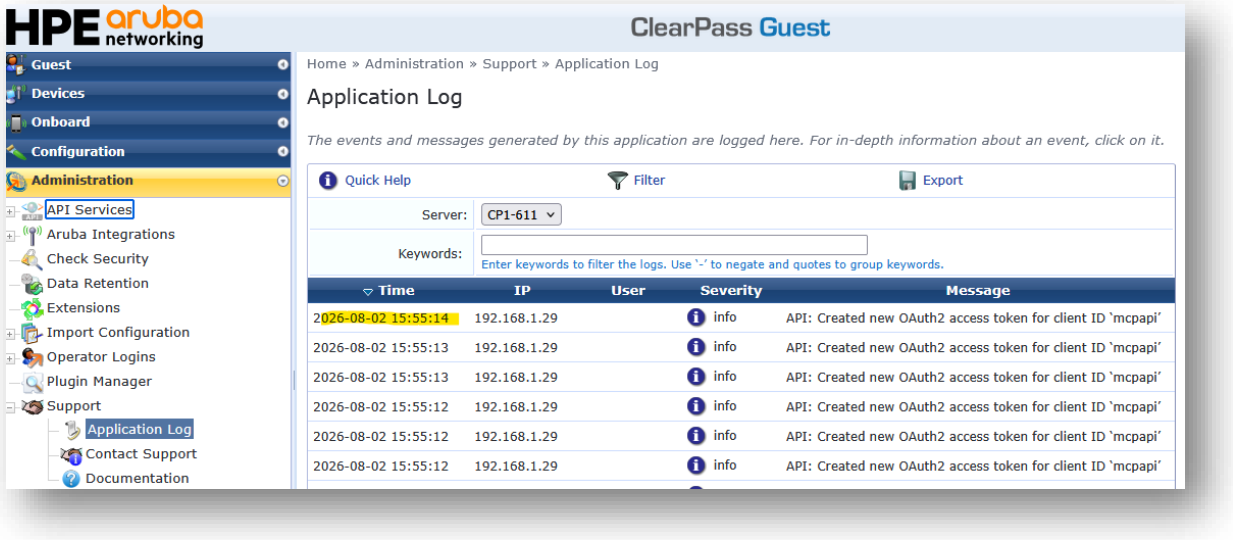
roles = ApiPolicyElements.get_role(login, limit=10)

for role in roles["_embedded"]["items"]:
    print(role["name"])
```

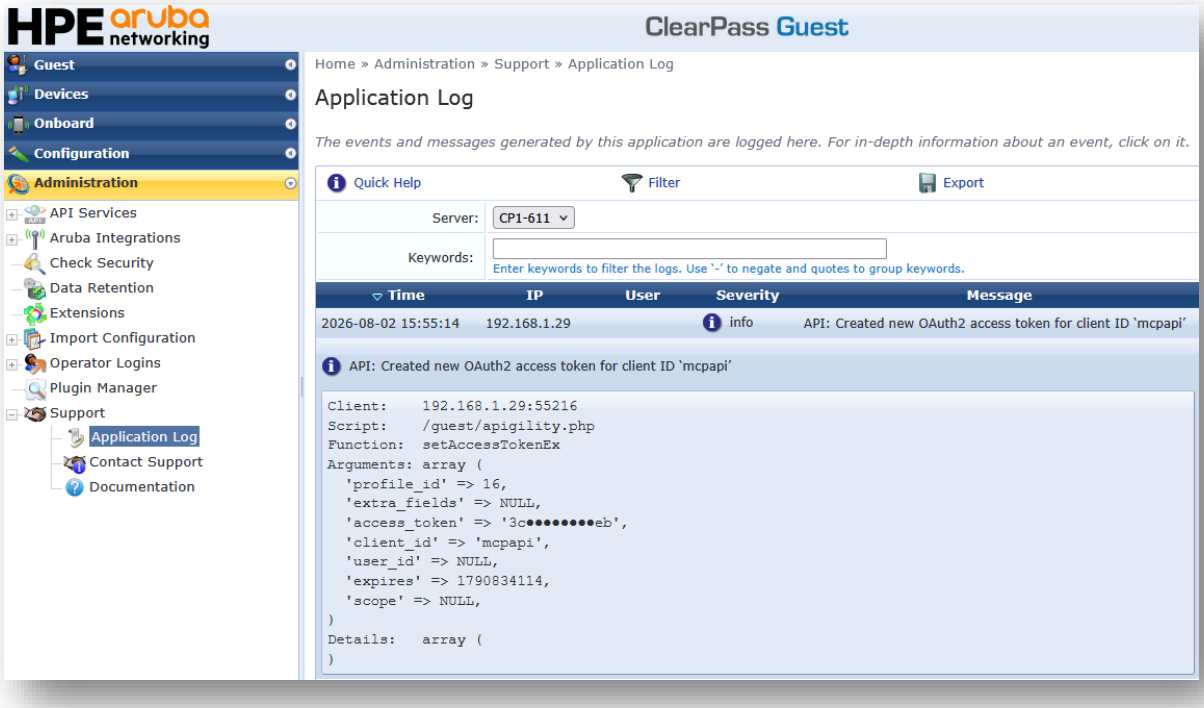
So now when you run this script, the output is as shown below. This proves that your API configuration was correct.

```
C:\scripts\ClearPass-pyclearpass>python clearpass-roles.py
[Guest]
[TACACS+ Read-only Admin]
[TACACS+ API Admin]
[TACACS+ Help Desk]
[TACACS+ Receptionist]
[TACACS+ Network Admin]
[TACACS+ Super Admin]
[Contractor]
[Other]
[Employee]
```

Here you can check the logs as well.



And by clicking on it you get the details.

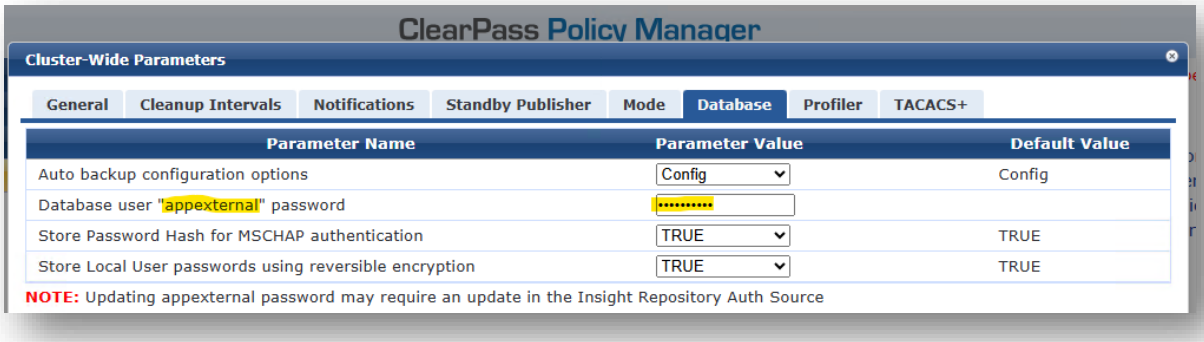


3.5 PostgreSQL Endpoint Profiling Access

From v1.6.0, CPV retrieves endpoint profiling information directly from the ClearPass PostgreSQL database. PostgreSQL access is therefore required for endpoint profiling and is configured during Initial Setup.

ClearPass provides the built-in appexternal database account for external PostgreSQL access. The External PostgreSQL Password is configured in ClearPass under:

Administration > Server Manager > Server Configuration > Cluster-Wide Parameters > Database



You will need this password later when completing the CPV Initial Setup. The PostgreSQL connection is validated before the Visualiser configuration is saved.

If PostgreSQL connectivity or authentication fails during application startup, endpoint profiling cache initialisation stops. CPV does not fall back to REST API endpoint profiling. The same appexternal account is used on port 5433 to read the current-hour Service Hit Counts from ClearPass Insight. If Insight cannot be reached, Hit Counts show n/a and the rest of CPV loads normally.

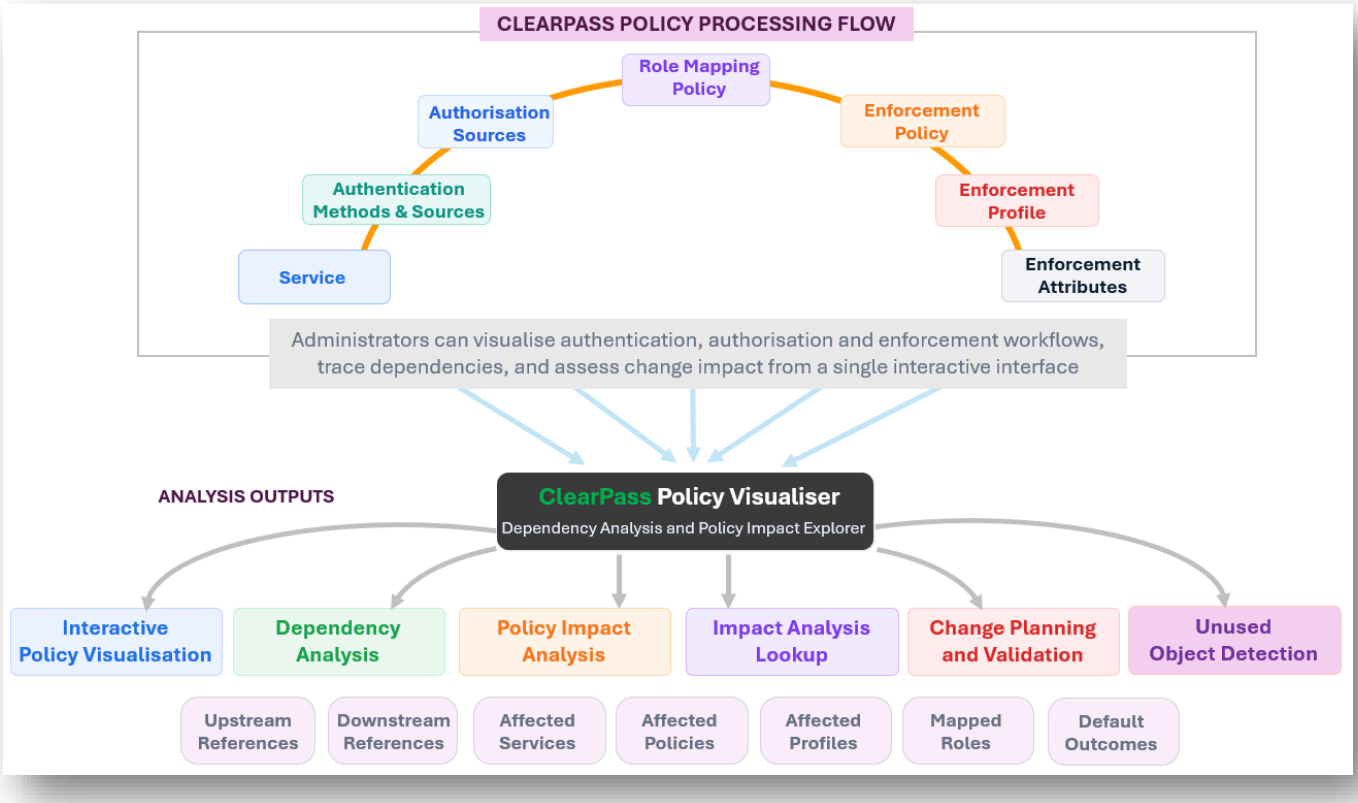
4 ClearPass Policy Visualiser Application

CPV is a Flask-based visualisation and analysis platform for ClearPass Policy Manager. The Visualiser consolidates policy relationships, dependencies and enforcement actions into an interactive interface, allowing administrators to understand ClearPass configuration without navigating multiple areas of Policy Manager.

The application uses ClearPass REST APIs secured with the OAuth 2.0 Client Credentials Grant for policy retrieval, analysis and optional configuration. Users authenticate against ClearPass using RADIUS, with application permissions derived from returned role attributes such as Aruba-User-Role.

CPV retrieves endpoint profiling data directly from the ClearPass PostgreSQL database using the built-in appexternal account. This provides efficient access to endpoint profiling attributes used by Endpoint Matching and Repository Search. PostgreSQL access is required for endpoint profiling, and the Visualiser no longer falls back to the REST API if PostgreSQL is unavailable.

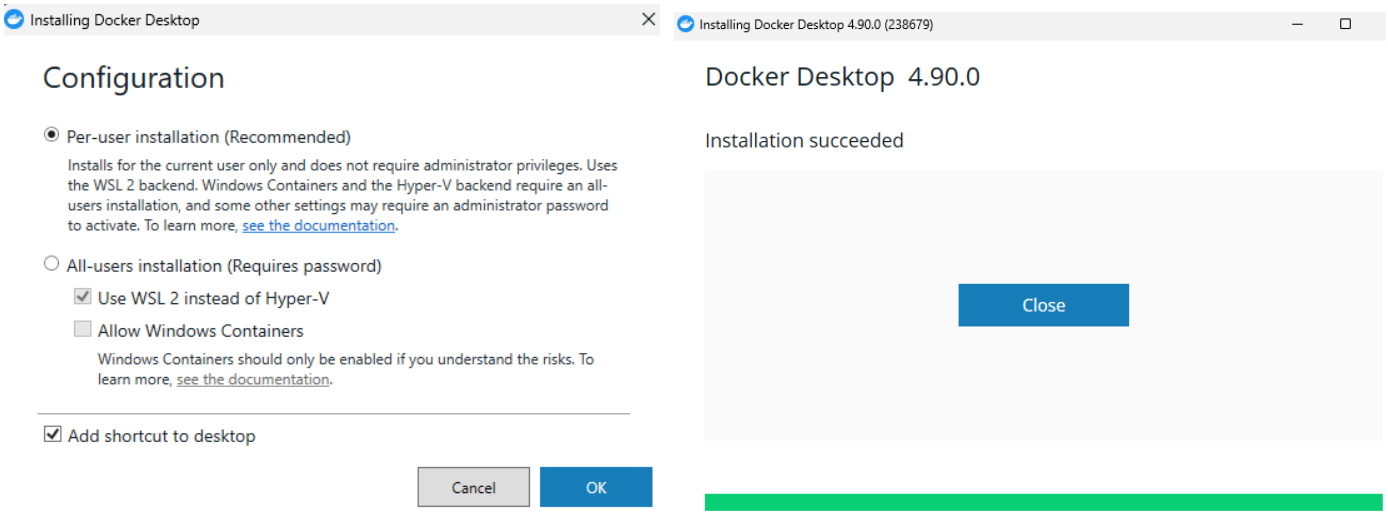
For deployment, Docker and Docker Compose are the recommended installation method. The container packages the Visualiser and its Python dependencies, while persistent storage retains the application configuration across container recreation. Native Python deployment remains available for environments where Docker is not used. From v1.7.0, one CPV instance can also connect to up to three ClearPass environments, each with its own REST API client, RADIUS server and PostgreSQL connection.



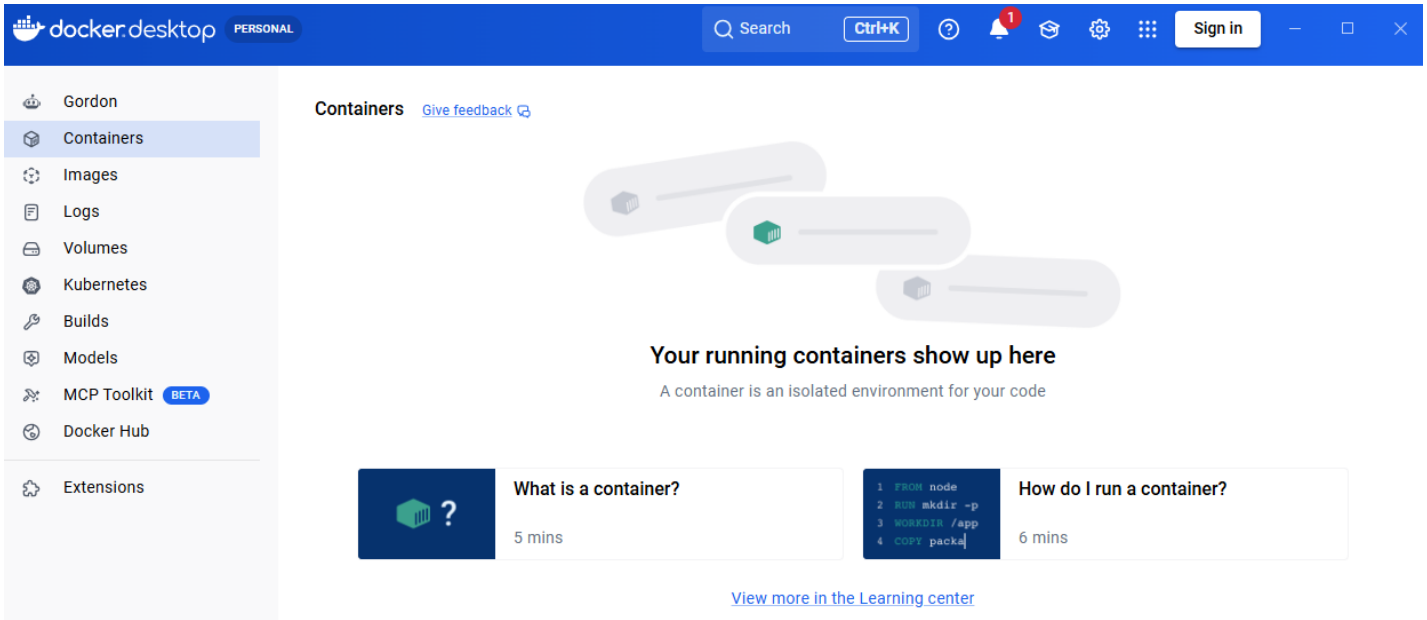
4.1 Installation

Docker is the recommended deployment method for CPV. It packages ClearPass Policy Visualiser and its Python dependencies into a container, removing the need to install and maintain a separate Python environment on the host.

For this technote, I'm using Docker Desktop on Windows. If you don't already have Docker Desktop installed, you can download it from the Docker website. Follow the Install Docker Desktop on Windows and make sure Docker Desktop is running before continuing with the CPV installation.



You can then run it as shown below.



Once Docker Desktop is installed, start it and wait for Docker to become ready. We can then verify WSL 2 and Docker Compose from the command line.

```
C:\PythonScriptingPostman>wsl --version
WSL version: 2.7.8.0
Kernel version: 6.18.33.1-1
WSLg version: 1.0.73.2
MSRDC version: 1.2.6676
Direct3D version: 1.611.1-81528511
DXCore version: 10.0.26100.1-240331-1435.ge-release
Windows version: 10.0.26100.8655

C:\PythonScriptingPostman>
C:\PythonScriptingPostman>docker compose version
Docker Compose version v5.5.1

C:\PythonScriptingPostman>
```

Now that Docker Desktop is installed and running, we can download CPV from GitHub.

Open a Windows Command Prompt and change to the directory where you want to store the Visualiser. Then clone the repository:

```
C:\PythonScriptingPostman>git clone https://github.com/ariyaparsa-dev/clearpass-policy-visualiser

Cloning into 'clearpass-policy-visualiser'...
```

```
remote: Enumerating objects: 248, done.
remote: Counting objects: 100% (248/248), done.
remote: Compressing objects: 100% (179/179), done.
remote: Total 248 (delta 132), reused 170 (delta 55), pack-reused 0 (from 0)
Receiving objects: 100% (248/248), 2.10 MiB | 12.42 MiB/s, done.
Resolving deltas: 100% (132/132), done.

C:\PythonScriptingPostman>
```

Change to the cloned project directory.

```
C:\PythonScriptingPostman>cd clearpass-policy-visualiser
C:\PythonScriptingPostman\clearpass-policy-visualiser>
```

Before starting the Visualiser, let's review the container timezone. The supplied compose.yaml uses the TZ environment variable to set the timezone used by the Visualiser container. If TZ is not defined, the container defaults to UTC. If UTC is appropriate for your environment, you can continue without making any changes.

If you want the Visualiser console and application timestamps to use your local timezone, create a Docker Compose **.env file** in the CPV project directory and specify the required timezone. For example, my Melbourne deployment uses:

```
TZ=Australia/Melbourne
```

The Docker Compose .env file is separate from the Visualiser .visualiser.env configuration stored in the persistent Docker volume. The Compose .env file is used for deployment settings such as the container timezone, while /data/.visualiser.env contains the ClearPass API, RADIUS and PostgreSQL configuration created during Initial Setup.

From the CPV project directory, start the application using Docker Compose:

```
docker compose up
```

```
C:\PythonScriptingPostman\clearpass-policy-visualiser>docker compose up
[+] up 1/1
 ! Image clearpass-policy-visua... pull access denied for clearpass-policy-visualiser, repository does not
exist or may require 'docker login' 1.9s
[+] Building 12.1s (13/13) FINISHED
 => [internal] load local bake definitions                                                    0.0s
 => => reading from stdin 626B                                                                0.0s
 => [internal] load build definition from Dockerfile                                          0.0s
 => => transferring dockerfile: 632B                                                          0.0s
 => [internal] load metadata for docker.io/library/python:3.11-slim                           0.0s
 => [internal] load .dockerignore                                                             0.0s
 => => transferring context: 188B                                                             0.0s
 => [1/6] FROM docker.io/library/python:3.11-
slim@sha256:9534e5a8e315485d4061ed659af0fd78a284c015f9b73661b41d6bab25604534                  0.0s
 => => resolve docker.io/library/python:3.11-
slim@sha256:9534e5a8e315485d4061ed659af0fd78a284c015f9b73661b41d6bab25604534                  0.0s
 => [internal] load build context                                                             0.0s
 => => transferring context: 3.01kB                                                           0.0s
 => CACHED [2/6] WORKDIR /app                                                                 0.0s
 => CACHED [3/6] COPY requirements.txt .                                                      0.0s
 => [4/6] RUN python -m pip install     --no-cache-dir     -r requirements.txt                 7.4s
 => [5/6] COPY . .                                                                            0.1s
 => [6/6] RUN groupadd         --gid 10001         visualiser     && useradd         --uid 10001         --gid
visualiser         --no-creat  0.3s
 => exporting to image                                                                        1.8s
 => => exporting layers                                                                       1.4s
 => => exporting manifest sha256:1caa59a6a1ef64bf907e47122f1cb6fc4cd76bb6eb97e0ffe50e2190ba10f72d     0.0s
 => => exporting config sha256:c55b82ed78c859196b4d4a124d035b29fcac7491a579164105236f5d74681413       0.0s
 => => exporting attestation manifest sha256:957f1a8243dec39bc10528d439cfb84a47d9766defac99a7903d30dda6138378
0.0s
 => => exporting manifest list sha256:b6b00bb740e2276d596c7e3e6c3f94a87d6c0d88dc9ea47776ffac14f4bd7a0f
0.0s
 => => naming to docker.io/library/clearpass-policy-visualiser:dev                            0.0s
[+] up 4/4acking to docker.io/library/clearpass-policy-visualiser:dev                         0.3s
 ✓ Image clearpass-policy-visualiser:dev       Built                                         15.1s
 ✓ Network clearpass-policy-visualiser_default Created                                        0.0s
 ✓ Volume clearpass-visualiser-data            Created                                        0.0s
 ✓ Container clearpass-policy-visualiser       Created                                        0.1s
Attaching to clearpass-policy-visualiser

clearpass-policy-visualiser  | 12-09-2026 06:34:16 INFO Generated new Flask session secret.
clearpass-policy-visualiser  | 12-09-2026 06:34:16 INFO Initial setup required. Missing ClearPass API
settings: CLEARPASS_API_URL, CLEARPASS_CLIENT_ID, CLEARPASS_CLIENT_SECRET
clearpass-policy-visualiser  | 12-09-2026 06:34:16 WARNING Initial setup is incomplete. Starting the web
server without initialising ClearPass caches.
```

```
clearpass-policy-visualiser  | 12-09-2026 06:34:16 INFO Starting ClearPass Policy Visualiser web server.
clearpass-policy-visualiser  | 12-09-2026 06:34:16 INFO Open http://127.0.0.1:5000 in your browser.
```

The initial image lookup may report that the image cannot be pulled before Docker Compose builds it locally from the included Dockerfile. In this example, the subsequent build completes successfully.

On the first start, Docker builds the CPV image, creates the required container and persistent storage, and starts the application.

Because this is a fresh installation, the Visualiser detects that no configuration exists and starts the web server without initialising the ClearPass caches. The console confirms that Initial Setup is required and displays the browser address for the Visualiser.

Now you can verify your timezone setting as well.

```
C:\PythonScriptingPostman\clearpass-policy-visualiser>docker exec clearpass-policy-
visualiser printenv TZ
Australia/Melbourne

C:\PythonScriptingPostman\clearpass-policy-visualiser>
```

Leave the Command Prompt running and open the address shown in the console in your browser. In this deployment, the Visualiser is available at http://127.0.0.1:5000.

4.2 Configuration

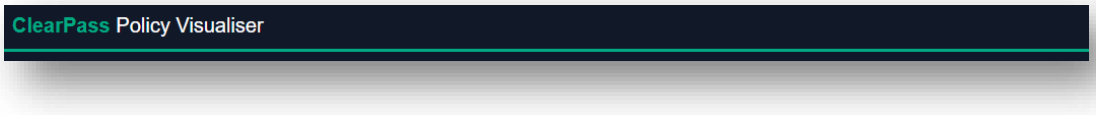
On a fresh installation, CPV automatically opens the Initial Setup page. Initial Setup collects the RADIUS authentication, ClearPass REST API and PostgreSQL endpoint profiling settings required by the Visualiser and validates the configuration before it is saved.

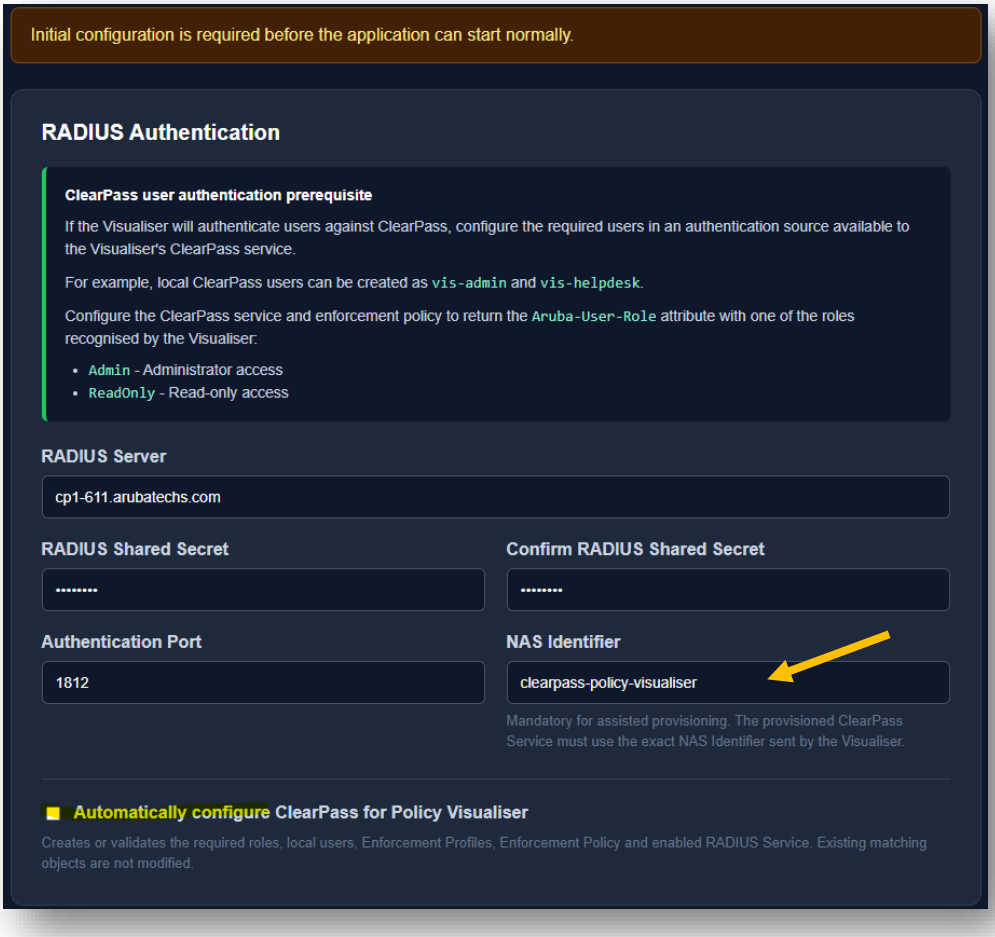
Browse to the address displayed in the console to open the Initial Setup workflow. The Visualiser authenticates users against ClearPass using RADIUS and expects ClearPass to return an Aruba-User-Role attribute containing one of the roles recognised by the Visualiser.

Here we have two options:

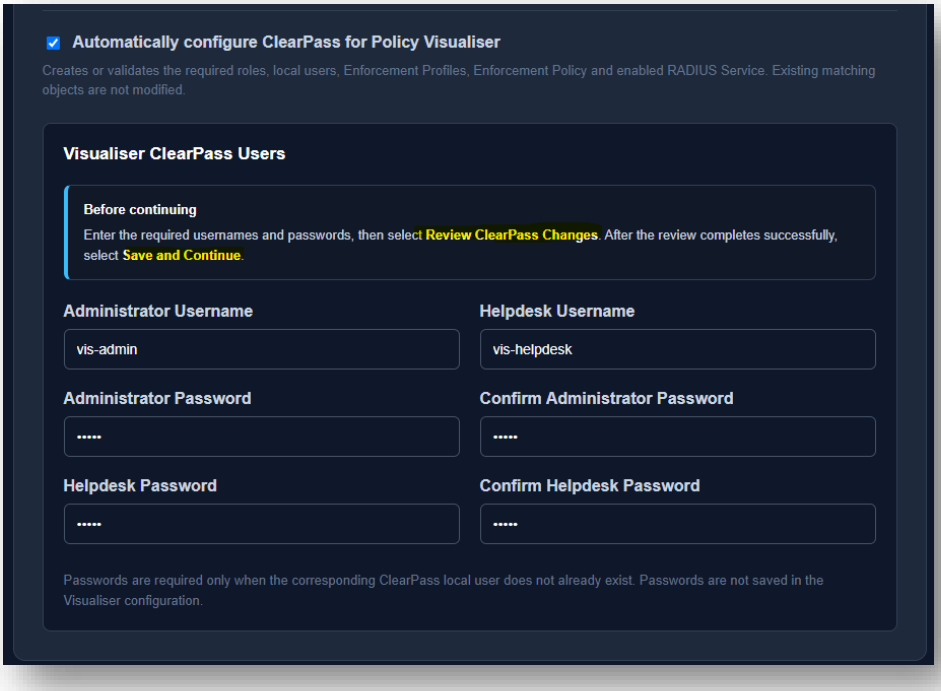
1. Use existing ClearPass service that already provide the required Visualiser authentication and role enforcement.
2. Allow the Visualiser to use the REST API to create or validate the required ClearPass objects.

For this example, I'll use the second option and allow the Visualiser to configure the required ClearPass objects automatically.

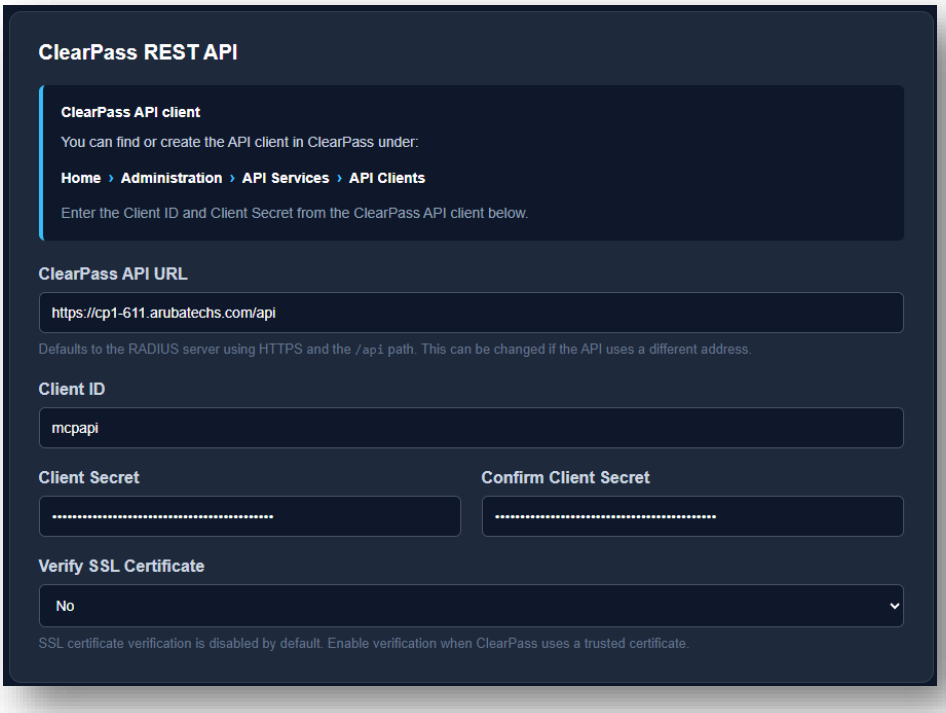




Selecting this option displays the Visualiser ClearPass Users section shown below.

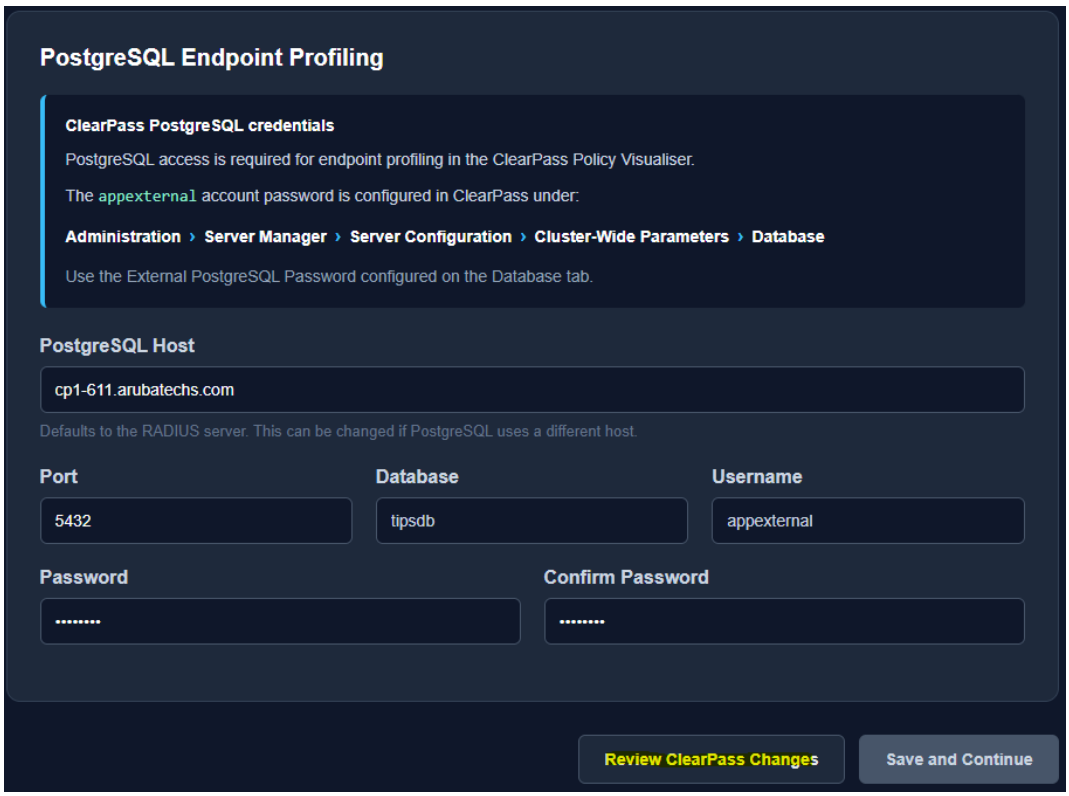


Next, scroll down and enter the ClearPass REST API settings.

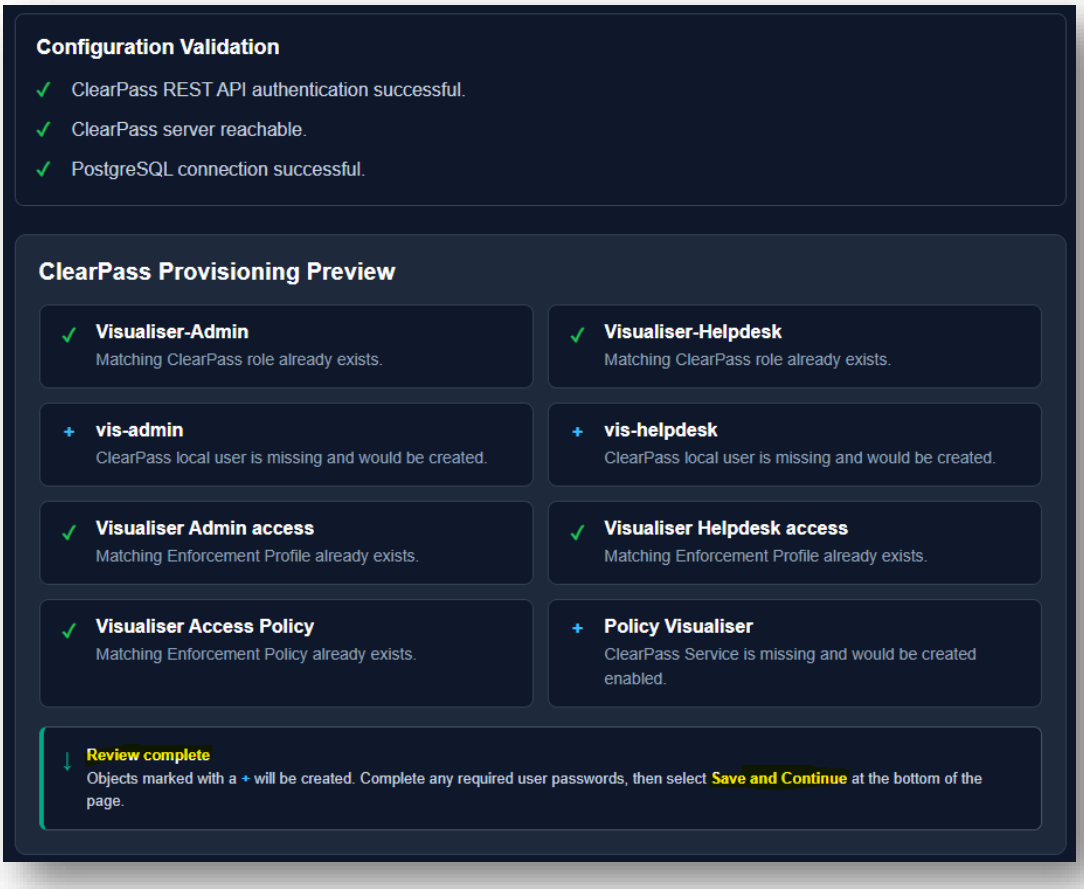


CPV uses direct PostgreSQL access for endpoint profiling. Unlike earlier versions, there is no profiling-source selection during Initial Setup. PostgreSQL access is required and the Visualiser does not fall back to REST API endpoint profiling. ClearPass provides the built-in appexternal account for external PostgreSQL access. The password for this account is configured in ClearPass under:

Administration > Server Manager > Server Configuration > Cluster-Wide Parameters > Database



Because automatic ClearPass configuration is selected, click Review ClearPass Changes. The Visualiser validates connectivity, REST API authentication and PostgreSQL connectivity before displaying the provisioning preview.

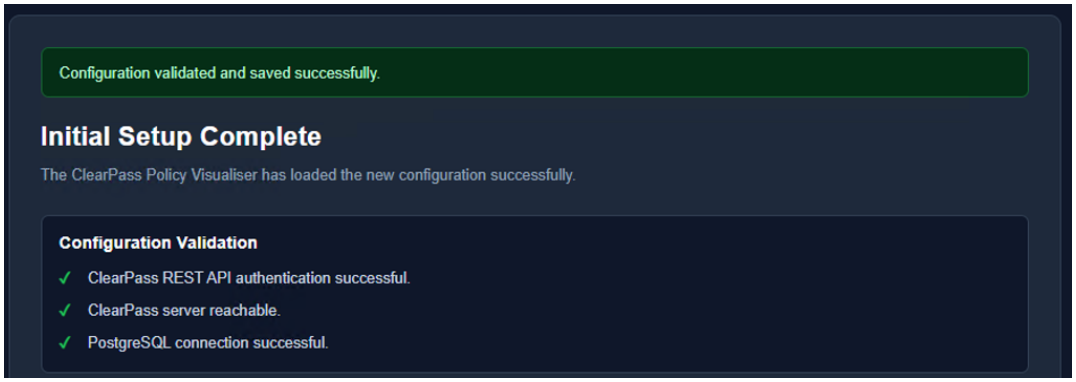


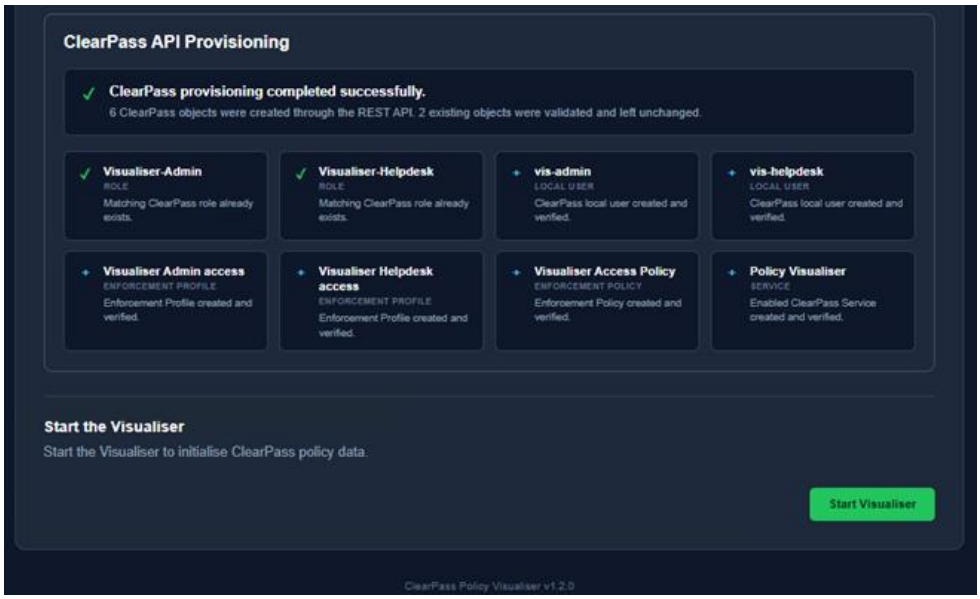
When you scroll to the bottom of the page, the “Save and Continue” button becomes active.



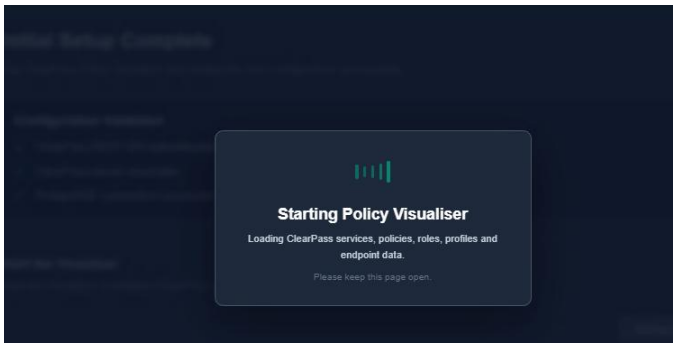
Once the required settings have been entered and the ClearPass change review has completed successfully, select Save and Continue.

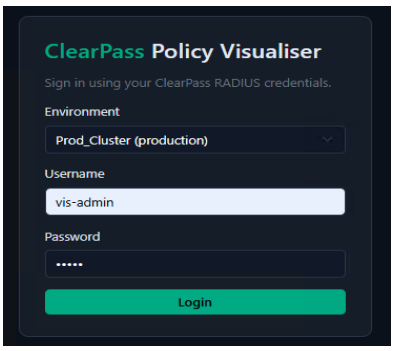
The Visualiser performs configuration validation before completing Initial Setup. If validation succeeds, the configuration is saved and the Initial Setup Complete page is displayed.





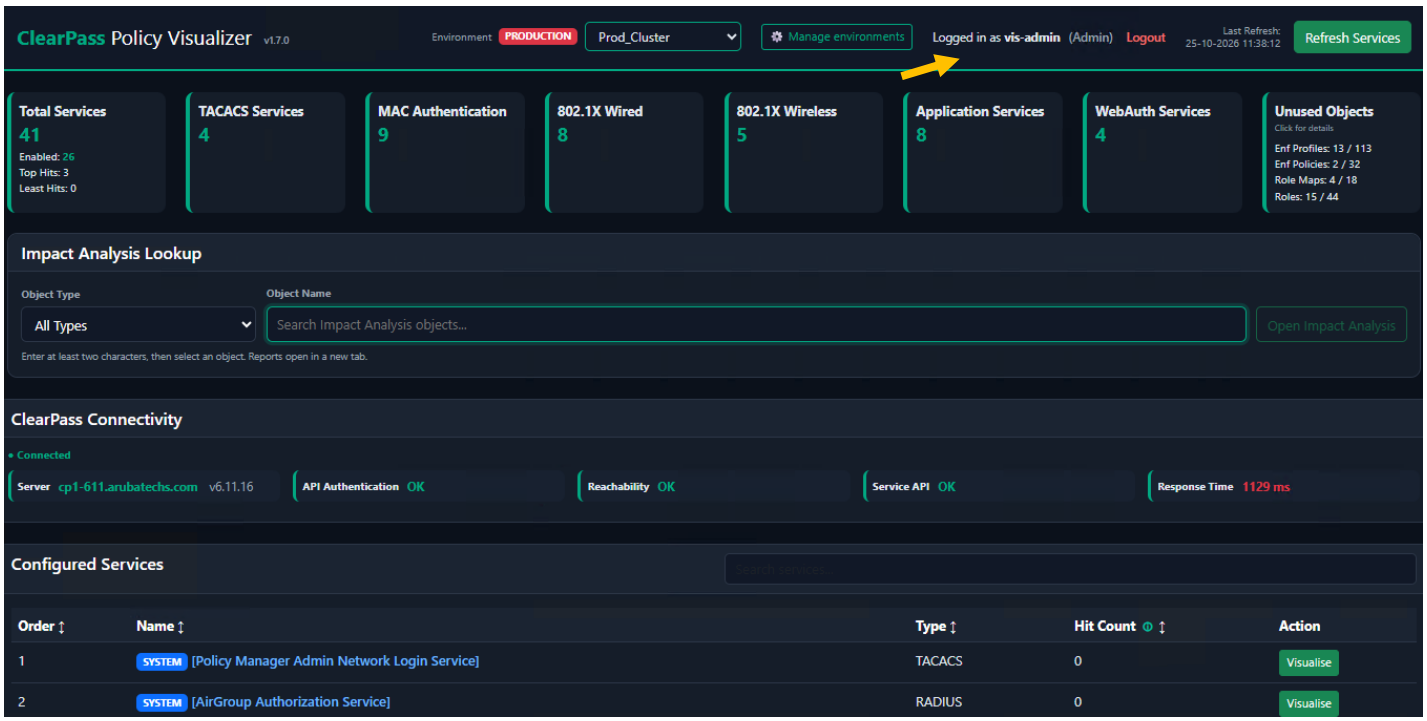
Now, we are at the stage that we can start the Visualiser by clicking on that button which will display the loading progress window shown below.





Once initialisation is complete, the Visualiser displays the login page. In this example, the vis-admin and vis-helpdesk users were created during Initial Setup, using the passwords entered earlier.

After successfully logging in as vis-admin, the Visualiser dashboard is displayed.

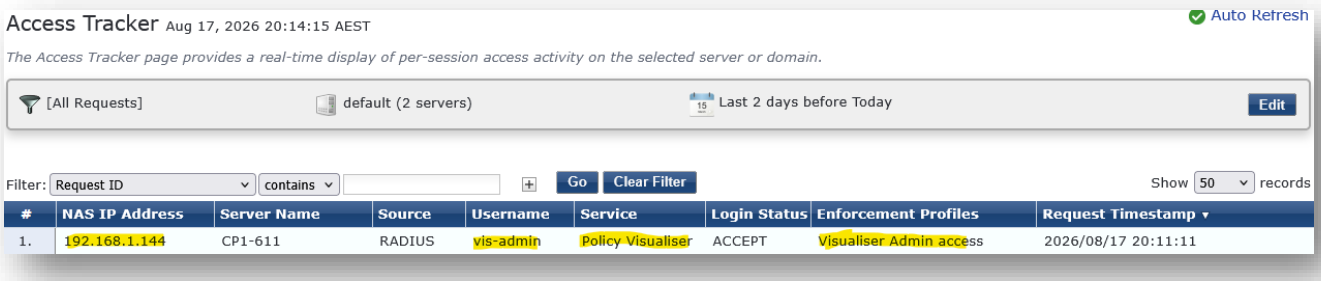


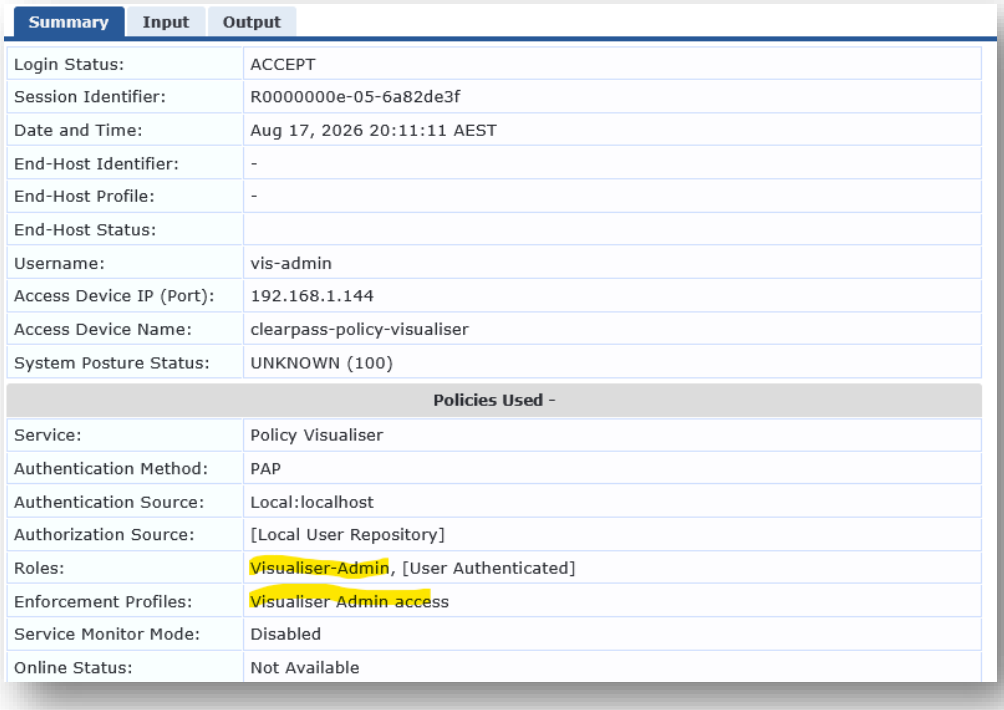
4.3 ClearPass Validation

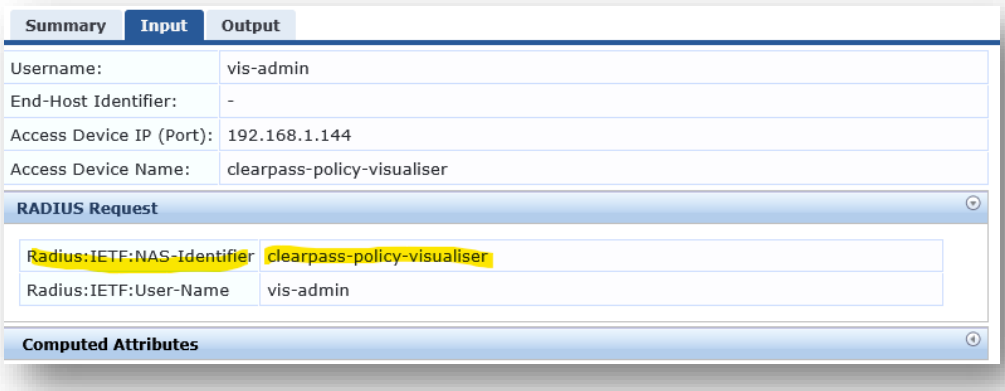
Now that the Visualiser has been configured and started, let's use ClearPass to validate the authentication and confirm the configuration created during Initial Setup.

First, open Access Tracker in ClearPass and locate the authentication request generated when logging in to ClearPass Policy Visualiser.

In this example, the vis-admin authentication is accepted and processed by the Policy Visualiser Service. The returned Visualiser-Admin role results in the Visualiser Admin access Enforcement Profile being applied.



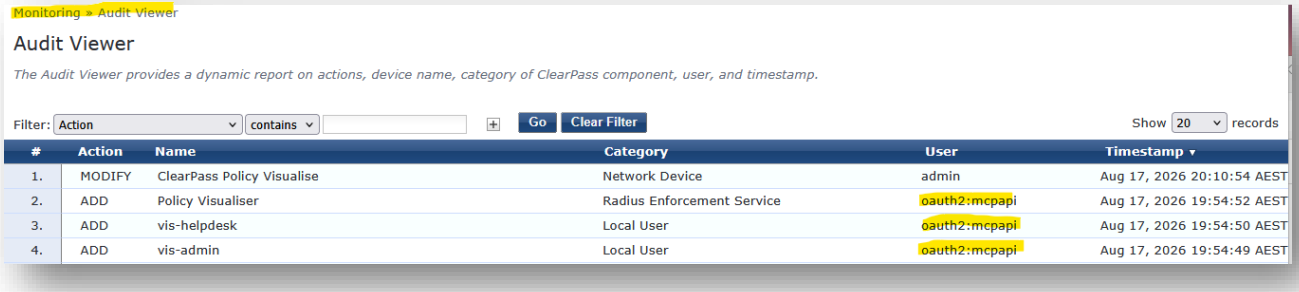




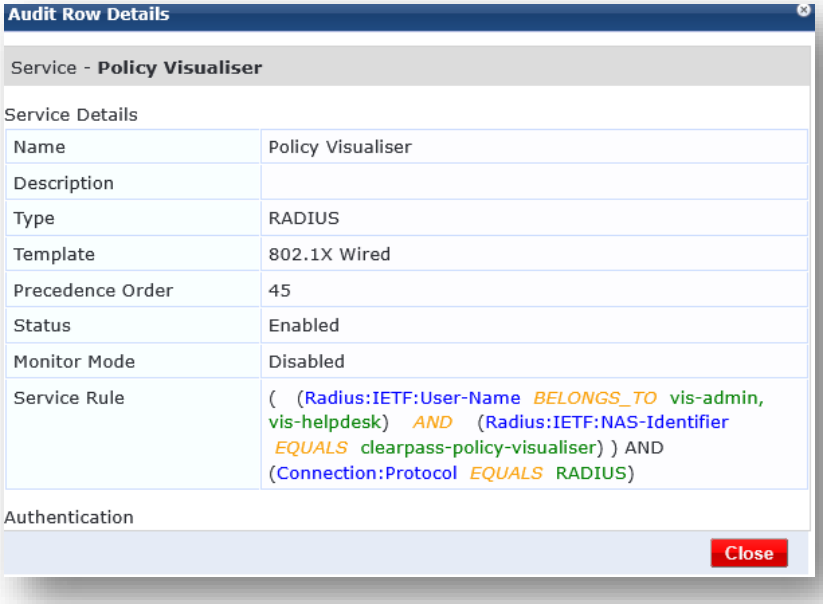


The RADIUS request also shows the NAS-Identifier sent by the Visualiser. In this example, the value is clearpass-policy-visualiser, which matches the NAS Identifier configured during Initial Setup.

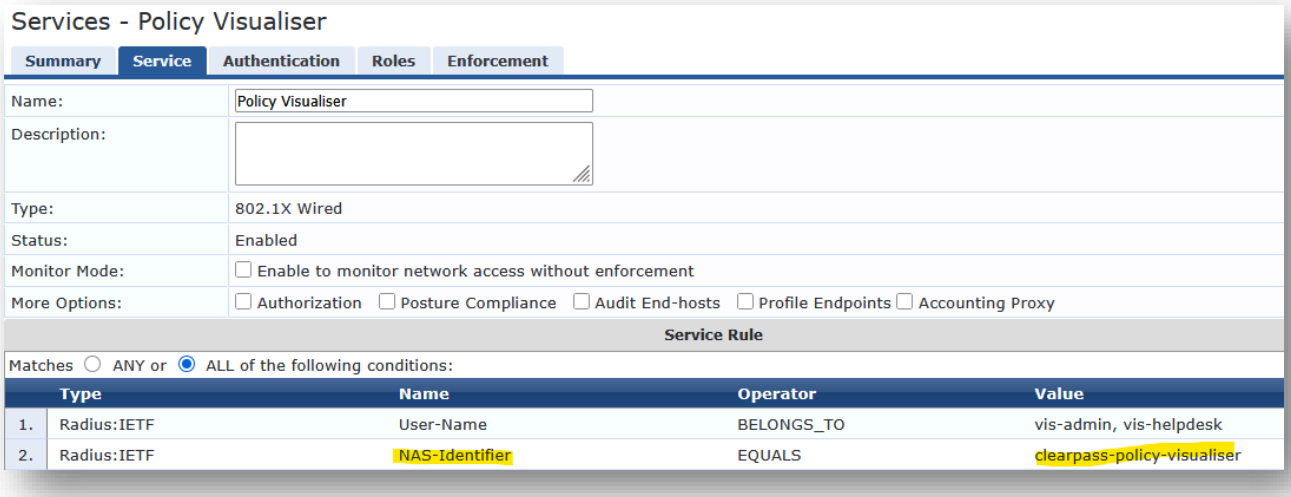
Next, open the ClearPass Audit Viewer. If API-assisted configuration was selected during Initial Setup, the Audit Viewer provides a record of the configuration changes performed by the Visualiser.

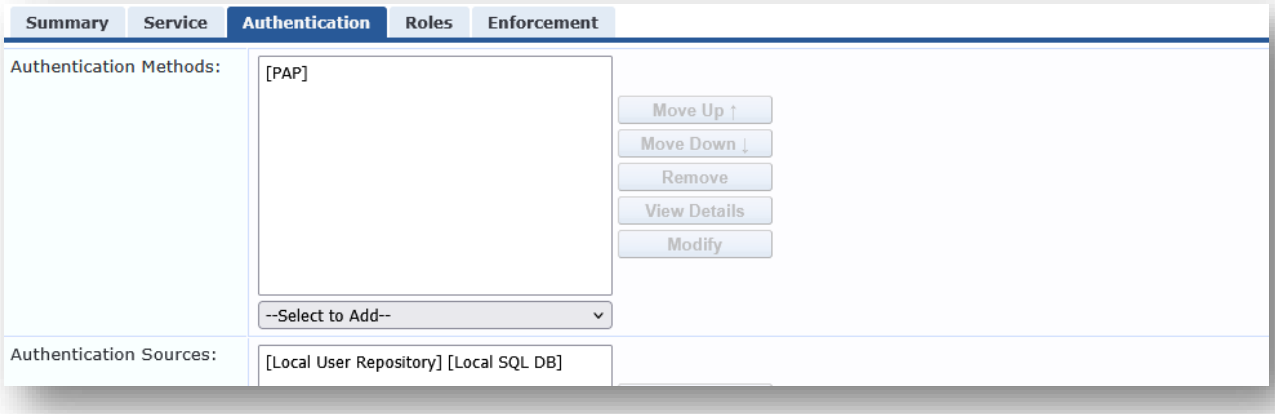


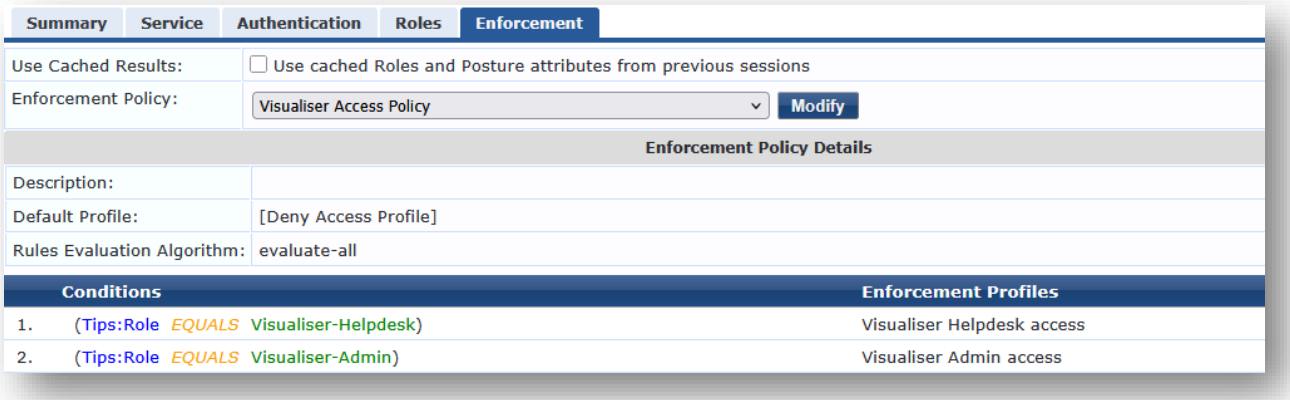
The example below shows the audit details associated with creation of the Policy Visualiser Service.



We can also review the Service created by the Visualiser. The newly provisioned Service is added to ClearPass without changing the order of existing Services.







The important thing is that this service matches on the NAS-Identifier being equal to “clearpass-policy-visualiser”.

PostgreSQL endpoint-profiling connectivity has already been validated as part of Initial Setup. A successful Configuration Validation confirms that the Visualiser can establish the PostgreSQL connection before the configuration is saved.

4.4 ClearPass Policy Visualiser Configuration

CPV stores the configuration created during Initial Setup in persistent Docker storage. The Docker Compose deployment uses the named volume clearpass-visualiser-data, which is mounted at /data inside the Visualiser container.

The **.visualiser.env** configuration file and the automatically generated **.flask_secret** file are stored in this persistent location rather than in the cloned project directory. This separates the application configuration from the container image and allows the configuration to remain available when the application container is recreated.

The **.visualiser.env** file contains the ClearPass REST API, RADIUS authentication and PostgreSQL endpoint profiling configuration entered during Initial Setup. Because this configuration includes sensitive connection credentials, it should be treated as sensitive application data.

```
# ClearPass Policy Visualiser
# Generated by Initial Setup

# ClearPass REST API
CLEARPASS_API_URL=https://clearpass.example.com/api
CLEARPASS_CLIENT_ID=visualiser-api
CLEARPASS_CLIENT_SECRET=<client-secret>
CLEARPASS_VERIFY_SSL=false

# RADIUS Authentication
RADIUS_SERVER=clearpass.example.com
RADIUS_AUTH_PORT=1812
RADIUS_SECRET=<radius-shared-secret>
RADIUS_NAS_IDENTIFIER=clearpass-policy-visualiser

# PostgreSQL Endpoint Profiling
CP_SQL_HOST=clearpass.example.com
CP_SQL_PORT=5432
CP_SQL_DATABASE=tipsdb
CP_SQL_USERNAME=appexternal
CP_SQL_PASSWORD=<external-postgresql-password>
CP_SQL_SSLMODE=prefer
CP_SQL_QUERY_FILE=sql/endpoint_profiles.sql
```

From v1.7.0, values are written in single quotes, for example **CLEARPASS_CLIENT_SECRET='<client-secret>'**. This keeps secrets containing spaces, # or quotes exactly as typed. A value containing a variable reference such as ${NAME} is refused. Files written by earlier versions without quotes still load normally.

The **.flask_secret** file contains the Flask session secret generated automatically by the Visualiser. Both files are owned by the Visualiser container user and are created with owner-only read/write permissions. The persisted configuration can be verified without displaying its contents by listing the /data directory inside the running container:

```
C:\PythonScriptingPostman\clearpass-policy-visualiser>docker exec clearpass-policy-visualiser ls -
la /data

total 16
drwxr-xr-x 2 visualiser visualiser 4096 Sep 12 08:39 .
drwxr-xr-x 1 root       root       4096 Sep 12 06:34 ..
-rw------- 1 visualiser visualiser   64 Sep 12 06:34 .flask_secret
-rw------- 1 visualiser visualiser  641 Sep 12 08:40 .visualiser.env

C:\PythonScriptingPostman\clearpass-policy-visualiser>
```

Docker retains the named volume independently of the Visualiser container. This means the application can be stopped and the container recreated without repeating Initial Setup. On subsequent starts, the Visualiser uses the existing configuration and proceeds directly with normal ClearPass data and cache initialisation.

Note that the clearpass-visualiser-data volume contains the persisted Visualiser configuration. Removing this volume removes the application's persisted configuration and may require Initial Setup to be completed again.

From v1.7.0, connection settings are normally changed on the **Environments page** and applied without a restart (see Managing Multiple ClearPass Environments). Changes made by editing the file directly still take effect only after the application is restarted. The same volume also holds /data/.visualiser.env.prev, a copy of the previous settings kept after an edit, and /data/environments/, which contains one settings file for each additional environment. Both contain secrets.

Stopping and Starting the Visualiser

When CPV is started using “docker compose up”, Docker Compose remains attached to the application and displays the Visualiser console output. To stop an attached Visualiser, press Ctrl+C in the terminal running Docker Compose. If you also want to remove the stopped container and its Compose network, run docker compose down as shown below.

```
Gracefully Stopping... press Ctrl+C again to force
Container clearpass-policy-visualiser Stopping
Container clearpass-policy-visualiser Stopped nable Watch   d Detach
```

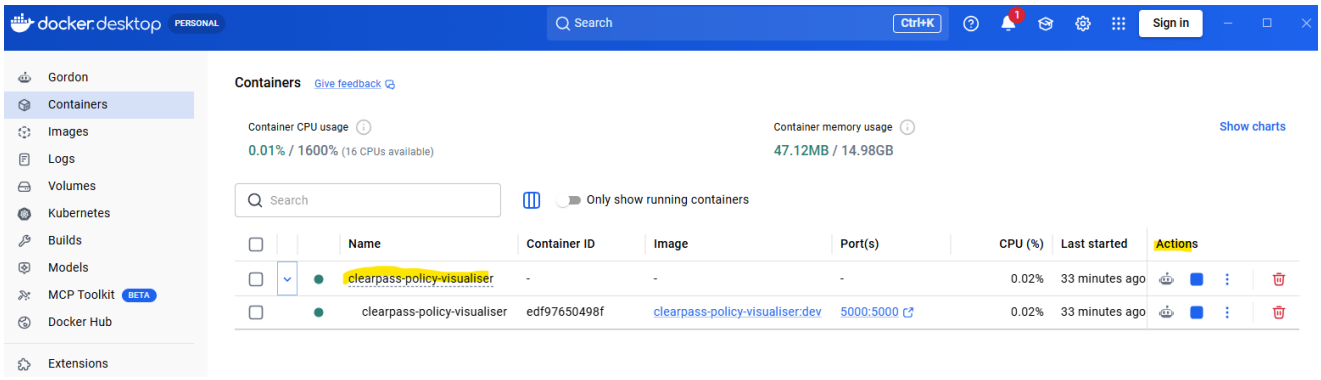
```
C:\PythonScriptingPostman\clearpass-policy-visualiser>
C:\PythonScriptingPostman\clearpass-policy-visualiser>docker compose down
[+] down 2/2
 ✓ Container clearpass-policy-visualiser       Removed                    0.1s
 ✓ Network clearpass-policy-visualiser_default Removed                    0.2s

C:\PythonScriptingPostman\clearpass-policy-visualiser>
```

Then to start the Visualiser again, from the CPV project directory run “docker compose up”.

The existing configuration stored in the clearpass-visualiser-data volume is reused when the Visualiser starts again, so Initial Setup does not need to be repeated.

You can also stop and start the CPV container from Docker Desktop, as shown below.



4.5 ClearPass Policy Visualiser Console

When CPV is started using Docker Compose in attached mode, the console displays timestamped operational messages as the application starts and builds its ClearPass data caches.

The console provides a useful operational view of the Visualiser startup process. Messages show the progress and outcome of ClearPass Service retrieval, health validation, PostgreSQL endpoint profiling, Role caching, policy-reference analysis, unused-object analysis and Impact Analysis lookup cache generation.

Counts and execution times provide additional diagnostic context and make it easier to confirm that each startup stage has completed successfully. Once cache initialisation is complete, the Visualiser starts the web server and displays the browser address. From v1.7.0, each line also carries the environment name in brackets, for example [Prod_Cluster] Roles cached: 67, so messages from different environments can be told apart. Additional environments are loaded the first time they are opened, so their startup messages appear later in the log.

To start the Visualiser and display the console output directly in the terminal, run the following command from the CPV project directory:

```
C:\PythonScriptingPostman\clearpass-policy-visualiser>
C:\PythonScriptingPostman\clearpass-policy-visualiser>docker compose up
[+] up 2/2
 ✓ Network clearpass-policy-visualiser_default Created                    0.0s
 ✓ Container clearpass-policy-visualiser       Created                    0.1s
Attaching to clearpass-policy-visualiser
clearpass-policy-visualiser  | 26-10-2026 16:19:53 INFO RADIUS_SERVER=cp1-611.arubatechs.com
clearpass-policy-visualiser  | 26-10-2026 16:19:53 INFO [Prod_Cluster]
============================================================
clearpass-policy-visualiser  | 26-10-2026 16:19:53 INFO [Prod_Cluster] ClearPass Policy Visualiser
v1.7.0 Starting
clearpass-policy-visualiser  | 26-10-2026 16:19:53 INFO [Prod_Cluster]
============================================================
clearpass-policy-visualiser  | 26-10-2026 16:19:53 INFO [Prod_Cluster] Loading ClearPass
services...
clearpass-policy-visualiser  | 26-10-2026 16:19:57 INFO [Prod_Cluster] Loading current-hour
Service hit counts from ClearPass Insight...
clearpass-policy-visualiser  | 26-10-2026 16:19:58 INFO [Prod_Cluster] Current-hour Service hit
counts loaded: 0 Services with hits.
clearpass-policy-visualiser  | 26-10-2026 16:19:58 INFO [Prod_Cluster] Service Hit Counts applied:
0 Services with current-hour hits.
clearpass-policy-visualiser  | 26-10-2026 16:19:58 INFO [Prod_Cluster] Services cached: 45
```

```
clearpass-policy-visualiser  | 26-10-2026 16:19:58 INFO [Prod_Cluster] Running initial health
check...
clearpass-policy-visualiser  | 26-10-2026 16:19:58 INFO [Prod_Cluster] Running health check...
clearpass-policy-visualiser  | 26-10-2026 16:20:00 INFO [Prod_Cluster] ClearPass version: 6.11.16
clearpass-policy-visualiser  | 26-10-2026 16:20:00 INFO [Prod_Cluster] Pre-loading endpoint
data...
clearpass-policy-visualiser  | 26-10-2026 16:20:00 INFO [Prod_Cluster] Pre-loading endpoint
profiling data using PostgreSQL...
clearpass-policy-visualiser  | 26-10-2026 16:20:00 INFO [Prod_Cluster] Endpoint profiling SQL load
complete: 426 rows in 0.175s
clearpass-policy-visualiser  | 26-10-2026 16:20:00 INFO [Prod_Cluster] Endpoint pre-load complete
using PostgreSQL: 426 fingerprint entries in 0.175s
clearpass-policy-visualiser  | 26-10-2026 16:20:00 INFO [Prod_Cluster] Building Role Cache...
clearpass-policy-visualiser  | 26-10-2026 16:20:00 INFO [Prod_Cluster] Roles cached: 68
clearpass-policy-visualiser  | 26-10-2026 16:20:00 INFO [Prod_Cluster] Building Enforcement
Profile Reference Cache...
clearpass-policy-visualiser  | 26-10-2026 16:20:04 INFO [Prod_Cluster] Retrieved 40 Enforcement
Policies from ClearPass
clearpass-policy-visualiser  | 26-10-2026 16:20:22 INFO [Prod_Cluster] Enforcement profile
reference cache built: 133 profiles, 40 total policies, 37 Service-assigned policies in 21.653s
clearpass-policy-visualiser  | 26-10-2026 16:20:22 INFO [Prod_Cluster] Building Role Mapping
Reference Cache...
clearpass-policy-visualiser  | 26-10-2026 16:20:23 INFO [Prod_Cluster] Building Role Reference
Cache...
clearpass-policy-visualiser  | 26-10-2026 16:20:35 INFO [Prod_Cluster] Retrieved 40 Enforcement
Policies from ClearPass
clearpass-policy-visualiser  | 26-10-2026 16:20:36 INFO [Prod_Cluster] Role reference cache built:
52 referenced Roles in 12.597s
clearpass-policy-visualiser  | 26-10-2026 16:20:36 INFO [Prod_Cluster] Role references cached: 52
clearpass-policy-visualiser  | 26-10-2026 16:20:36 INFO [Prod_Cluster] Building Unused Objects
Cache...
clearpass-policy-visualiser  | 26-10-2026 16:20:42 INFO [Prod_Cluster] Building Impact Analysis
Lookup Cache...
clearpass-policy-visualiser  | 26-10-2026 16:20:43 INFO [Prod_Cluster] Retrieved 161 Enforcement
Profiles from ClearPass
clearpass-policy-visualiser  | 26-10-2026 16:20:44 INFO [Prod_Cluster] Retrieved 40 Enforcement
Policies from ClearPass
clearpass-policy-visualiser  | 26-10-2026 16:20:45 INFO [Prod_Cluster] Impact Analysis lookup
objects cached: 289
clearpass-policy-visualiser  | 26-10-2026 16:20:45 INFO [Prod_Cluster] Cache initialisation
complete.
clearpass-policy-visualiser  | 26-10-2026 16:20:45 INFO [Prod_Cluster] Services: 45
clearpass-policy-visualiser  | 26-10-2026 16:20:45 INFO [Prod_Cluster] Role Mapping Policies: 16
clearpass-policy-visualiser  | 26-10-2026 16:20:45 INFO [Prod_Cluster] Enforcement Profiles: 133
clearpass-policy-visualiser  | 26-10-2026 16:20:45 INFO [Prod_Cluster] Impact Analysis lookup
objects: 289
clearpass-policy-visualiser  | 26-10-2026 16:20:45 INFO [Prod_Cluster] Last Refresh: 26-10-2026
16:19:53
clearpass-policy-visualiser  | 26-10-2026 16:20:45 INFO [Prod_Cluster] Starting ClearPass Policy
Visualiser web server.
clearpass-policy-visualiser  | 26-10-2026 16:20:45 INFO [Prod_Cluster] Open http://127.0.0.1:5000
in your browser
```

The startup messages are displayed in the order in which the Visualiser prepares its working data. In this example, all startup stages complete successfully before the web server is started.

- Loading ClearPass services retrieves the configured ClearPass Services and populates the Service cache used by the Visualiser. The console reports the number of Services cached when this stage completes.
- Running initial health check verifies the configured ClearPass connectivity before the remaining caches are built.
- Endpoint profiling data is loaded directly from ClearPass PostgreSQL. The console shows when the PostgreSQL profiling load begins, the number of endpoint fingerprint records returned and the time required to populate the endpoint profiling cache.

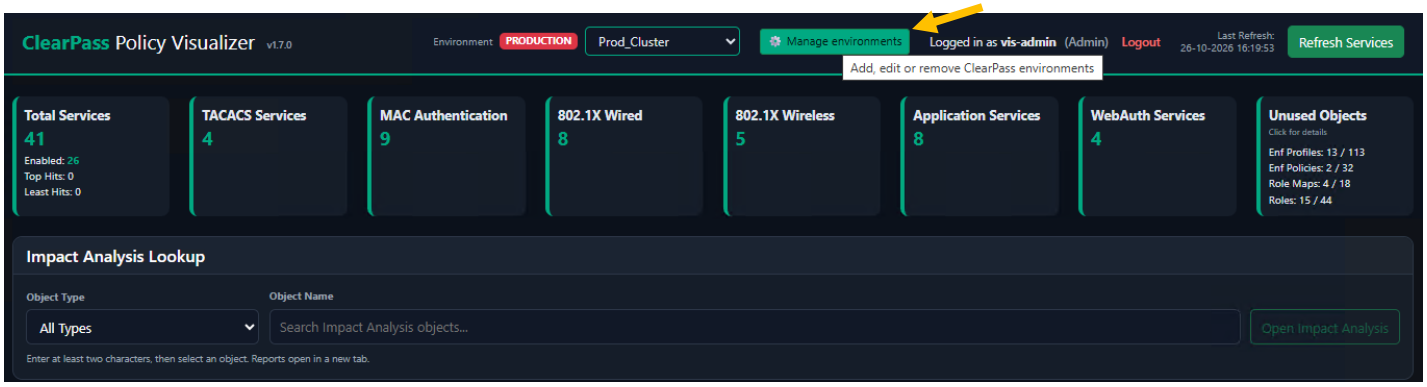
```
INFO [Prod_Cluster] Endpoint profiling SQL load complete: 426 rows in 0.175s
INFO [Prod_Cluster] Endpoint pre-load complete using PostgreSQL: 426 fingerprint entries in
0.175s
```

In this example, 426 endpoint profiling records were loaded successfully using PostgreSQL. The number of records and load time will vary between ClearPass environments.

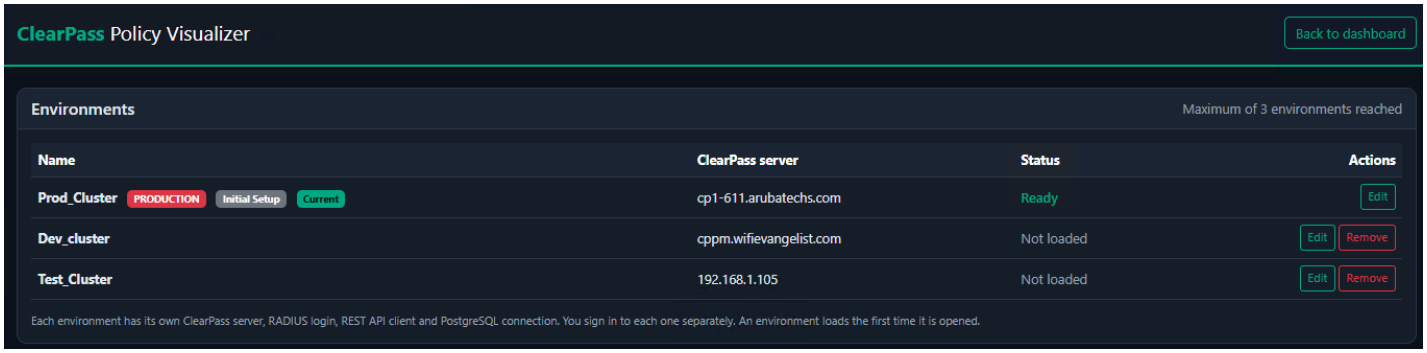
- The Visualiser builds several additional caches used for policy visualisation, dependency analysis and object lookups. These include the Role Cache, Enforcement Profile Reference Cache, Role Mapping Reference Cache and Role Reference Cache.
- Building Unused Objects Cache prepares the relationship information used by the Unused Objects capability described later in this technote.
- Building Impact Analysis Lookup Cache prepares the object index used by the dashboard Impact Analysis Lookup. The console reports the number of lookup objects cached when this stage completes.
- Cache initialisation complete. confirms that the startup cache-building sequence has finished. The console then displays summary counts and starts the CPV web server.
- When startup completes successfully, the console displays the browser address for the Visualiser.

4.6 Managing Multiple ClearPass Environments

From v1.7.0, one CPV instance can manage up to three ClearPass environments, for example Production, Test and Dev. The environment created by Initial Setup is the initial environment. Each environment has its own REST API client, RADIUS server, PostgreSQL connection and cached data, so analysis in one environment never uses data from another.

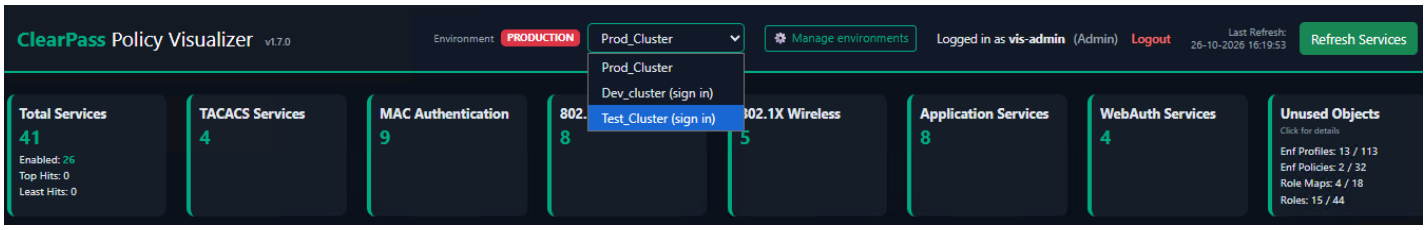


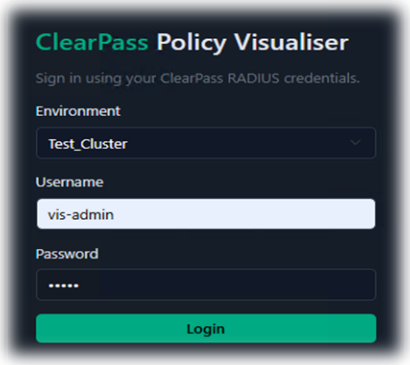
Administrators open the Environments page by selecting **Manage environments** on the dashboard. From this page you can: **Add** an environment (up to three in total), **Edit** an environment's connection settings, including the initial environment. You can further **Rename** an environment, or mark it as production to show a red PRODUCTION badge and finally **Remove** an environment other than the initial one.

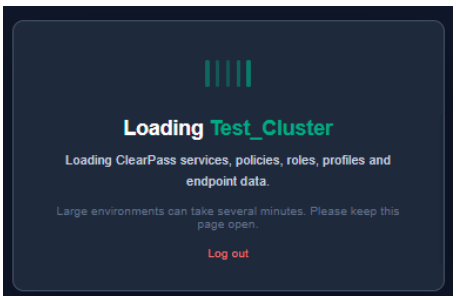


Adding an environment. The Add environment form follows the same layout as Initial Setup, with RADIUS, REST API and PostgreSQL sections. The API URL and PostgreSQL host default to the RADIUS server as you type. Select **Test connection and add** to validate the settings. The environment is saved only if validation passes. As in Initial Setup, you can select **Automatically configure ClearPass** and use **Review ClearPass Changes** to create any missing Visualiser objects in the new ClearPass.

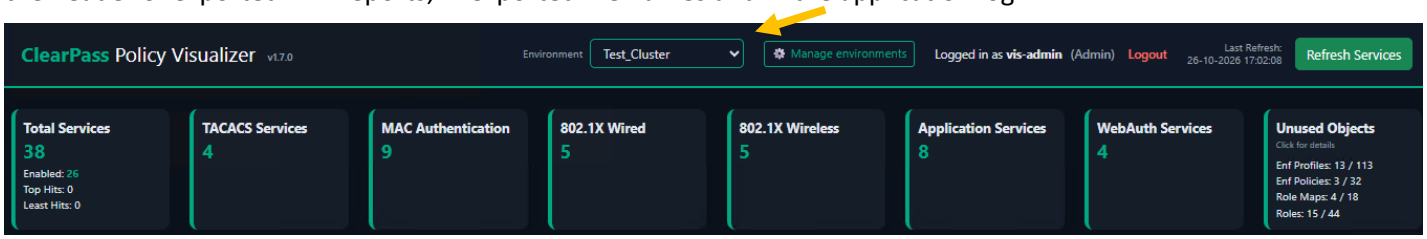
Signing in and switching. Each environment uses its own RADIUS server, so users sign in to each environment separately, and a user can be Admin in one environment and ReadOnly in another. Use the Environment selector in the navbar to switch. Additional environments load in the background the first time they are opened, with a loading panel. If loading fails, CPV shows Unable to load with a Try again button, and the other environments are not affected.







Knowing which environment you are in. The active environment is shown on every page and Impact Analysis report, in the header of exported PDF reports, in exported file names and in the application log.



Editing the initial environment. Changes are tested, saved to /data/.visualiser.env and applied without a restart. The previous settings are kept as /data/.visualiser.env.prev. If a change to the RADIUS settings fails its test, CPV asks you to tick Save these RADIUS settings anyway before saving, because a wrong RADIUS setting can stop everyone signing in to that environment. Delete .visualiser.env.prev when you no longer need it, because it contains secrets.

4.7 Connection Diagnostics

When a connection check fails, Initial Setup and the Environments page explain the likely cause in plain language rather than only reporting a failure. For example, CPV reports when:

- the server name does not resolve, nothing answers, or the host is up but the port is blocked or refused
- the API URL does not end in /api, uses http:// or port 80, or the certificate is not trusted
- PostgreSQL rejects the appexternal password, or does not allow the CPV host's address

Amber notes, which do not block saving, flag a PostgreSQL or RADIUS host that differs from the API host. When adding an environment, CPV also checks that Insight is reachable on port 5433.

RADIUS check. CPV sends one PAP Access-Request for the user cpv-connectivity-test with a random password. A signed reply proves that the RADIUS server is reachable, that the CPV host is a defined Network Device and that the shared secret matches. The request appears in Access Tracker as a rejected authentication, usually with Service Categorization failed. This is expected, because the Visualiser Service only matches the Visualiser users.

With the Visualiser installed, configured and successfully initialised, we can now move on to its operational capabilities and explore the dashboard, dependency graphs, Impact Analysis, endpoint matching and unused-object analysis.

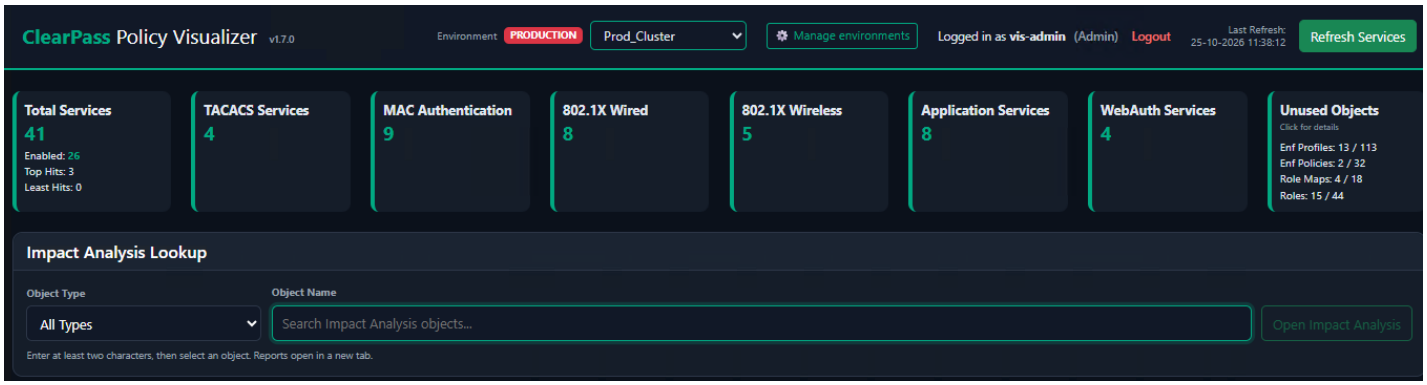
5 ClearPass Policy Visualiser Operation

This section introduces the main operational capabilities of ClearPass Policy Visualiser. It explains how to use the dashboard, explore interactive dependency graphs, search for policy objects, perform Impact Analysis, inspect matching endpoints and identify unused configuration objects.

The examples demonstrate how administrators can trace policy relationships, validate processing logic and investigate the potential scope of configuration changes from a single interface.

5.1 Dashboard

When you log in using vis-admin or vis-helpdesk account, the dashboard provides an immediate overview of the ClearPass environment. At the top of the dashboard, you can see the total number of configured Services, how many are enabled, and a breakdown by Service type. The summary includes TACACS, MAC Authentication, wired and wireless 802.1X, Application and WebAuth Services. The Unused Objects card is also displayed in this area.

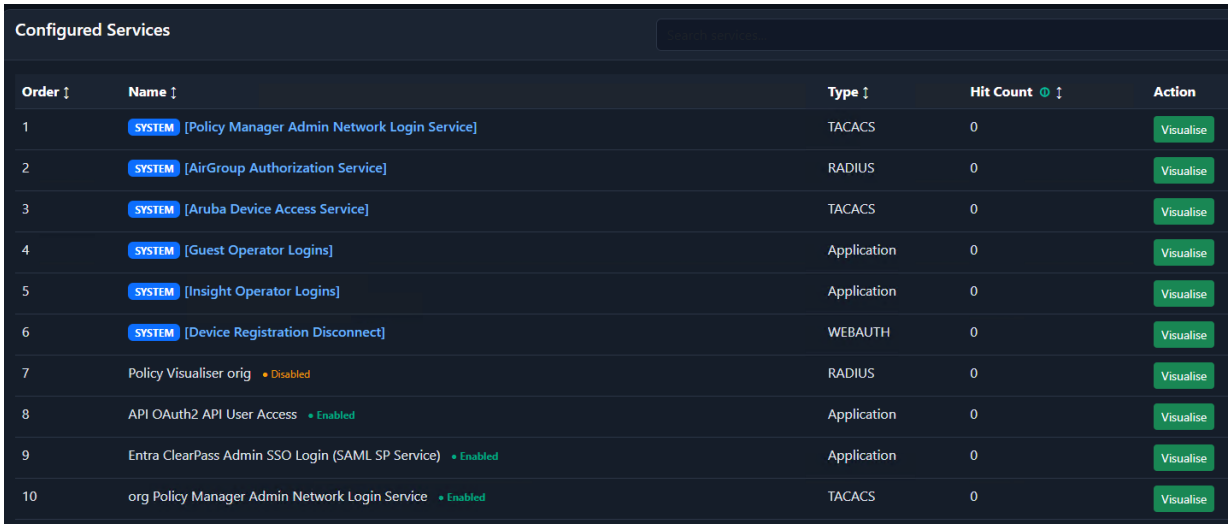


The dashboard also displays the current ClearPass connectivity status, including API authentication, reachability, Service API status and response time. In my lab environment, the API response time is more than one second, but this will vary depending on the ClearPass deployment, network path and available resources. From v1.7.0, the connectivity card also shows the ClearPass version of the active environment. When more than one environment is configured, the navbar shows an Environment selector, and administrators also see a Manage environments button.

The Impact Analysis Lookup provides a quick way to find an Enforcement Profile, Enforcement Policy, Role Mapping Policy or Role. You can search across all supported object types or filter the results by object type. Once you select an object, you can open its Impact Analysis report in a new browser tab.

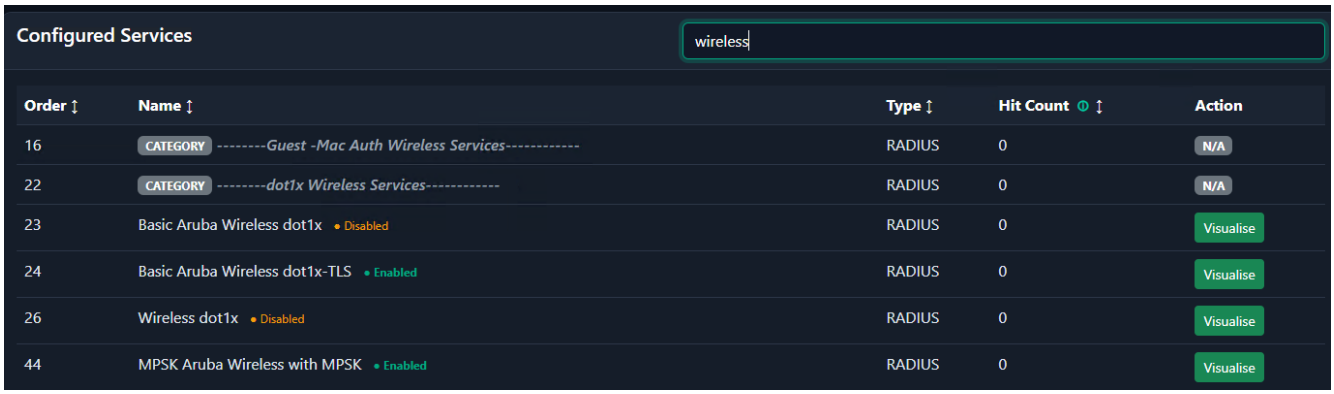
The Refresh Services button forces CPV to retrieve the latest ClearPass data and rebuild its caches. This is useful after making configuration changes in ClearPass. The refresh includes Services, Roles, policy references, unused-object analysis, Service Hit Counts and the Impact Analysis lookup index. It refreshes only the active environment.

As you move down the page you should see all the configured services.



The Configured Services table lists each discovered Service, its type, enabled or disabled state, and a Visualise button for opening the interactive dependency graph.

You can use the Service search field to filter the list quickly. In the example below, I've searched for wireless, which displays each Service containing that word in its name.

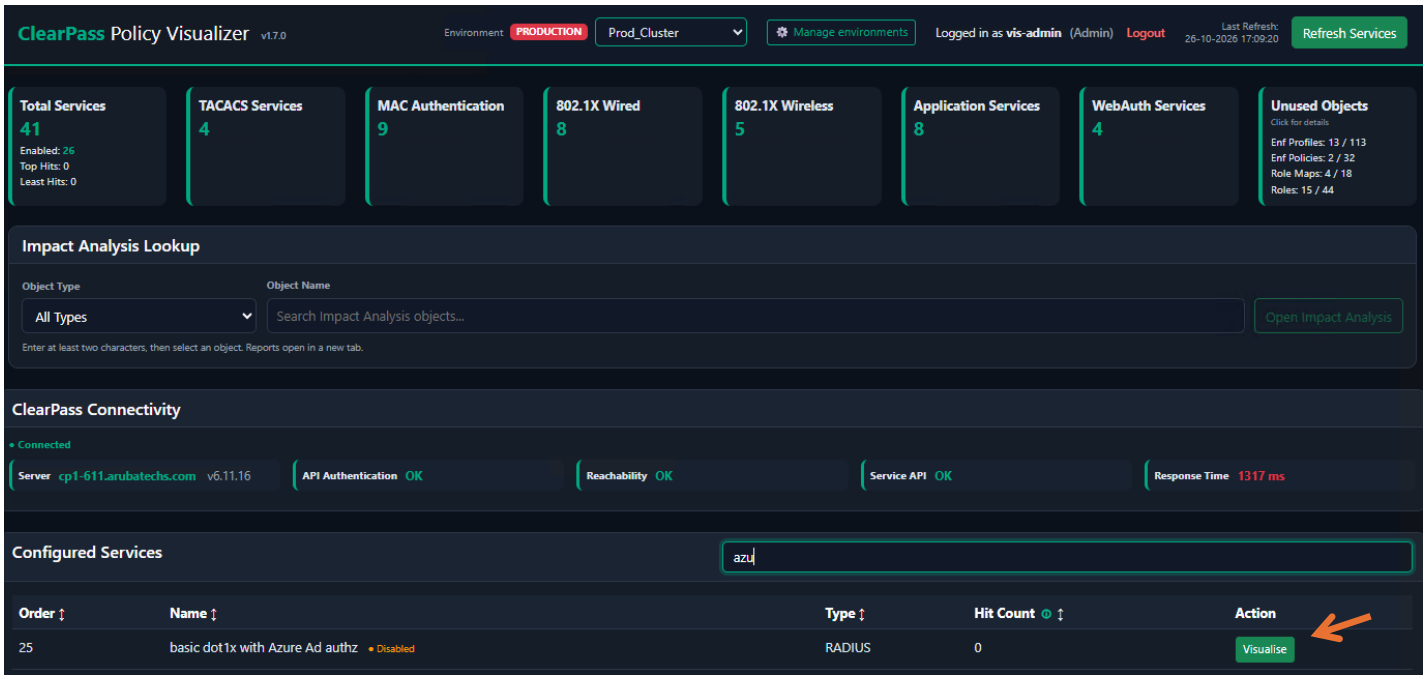


If, like me, you use names beginning with several hyphens to create visual categories in the ClearPass Service list, the Visualiser recognises them as category entries. These entries are labelled CATEGORY and are not presented as Services that can be visualised.

5.2 Visualisation and Interactive Dependency Graph

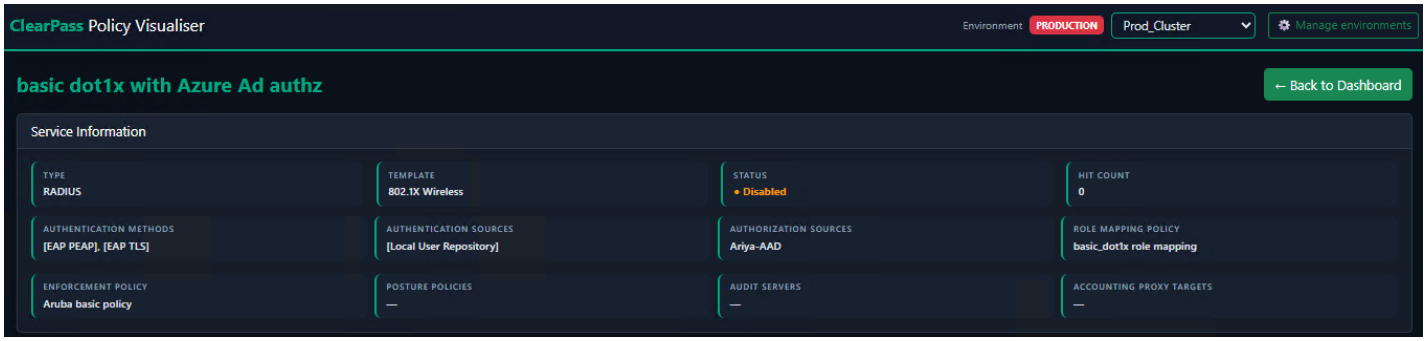
Let's search for a Service and open its interactive dependency graph.

The Configured Services table lists each discovered Service, its type, enabled or disabled state, and a Visualise button. You can use the search field to filter the list quickly. For this walkthrough, I'll search for my Azure-related Service and select Visualise.



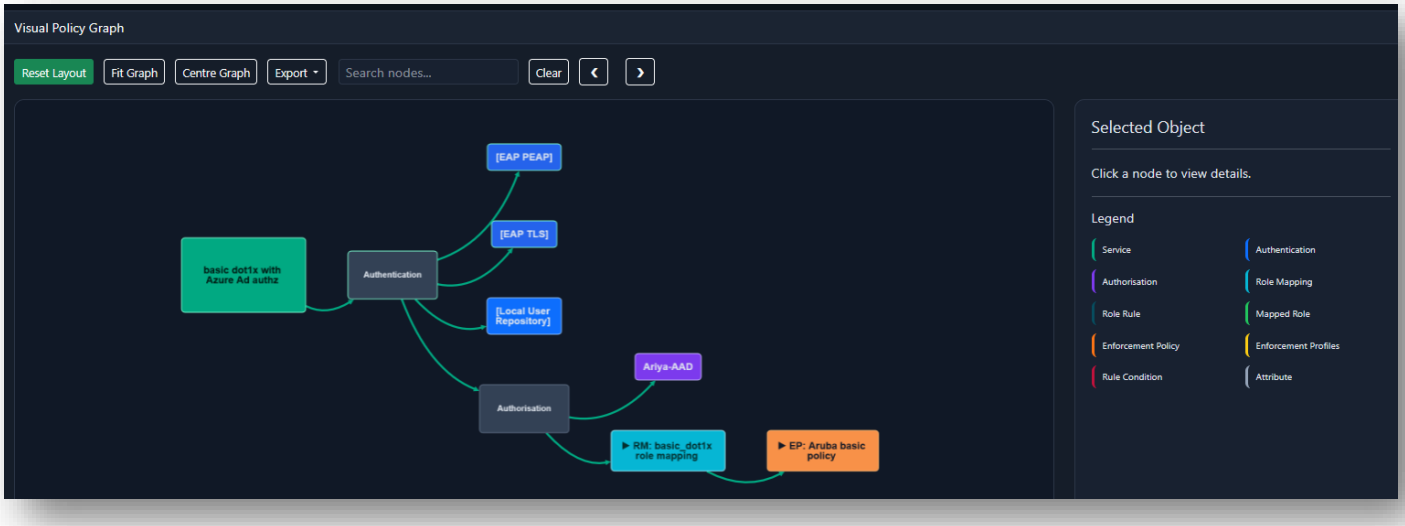
The Service page opens with a summary of the selected configuration. This includes the Service type, template, enabled or disabled state, authentication methods, authentication and authorisation sources, Role Mapping Policy and Enforcement Policy.

This summary provides a quick way to confirm that the expected policy components are associated with the Service before exploring the graph.



The Visual Policy Graph appears below the Service summary. The initial view presents the main authentication, authorisation, Role Mapping and Enforcement relationships without displaying every underlying rule and profile.

The graph includes controls for resetting the layout, fitting and centring the graph, exporting the current view, and searching for individual nodes. Selecting a node displays its details in the Selected Object panel on the right.

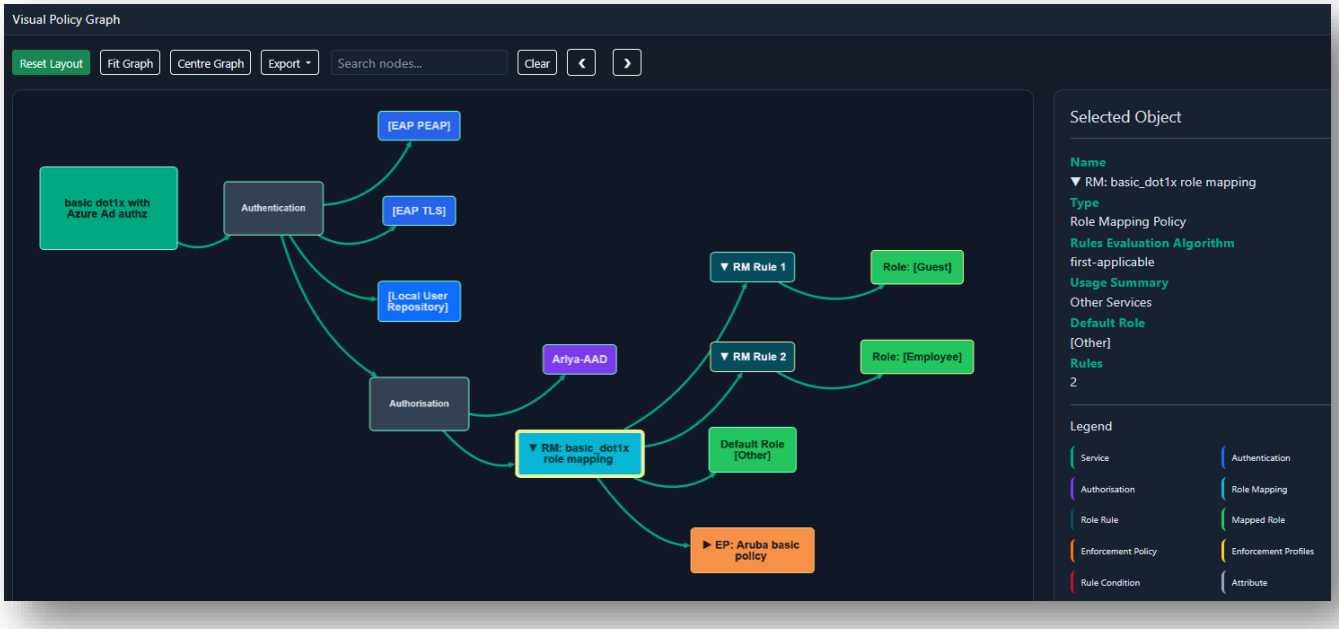


The interactive graph provides:

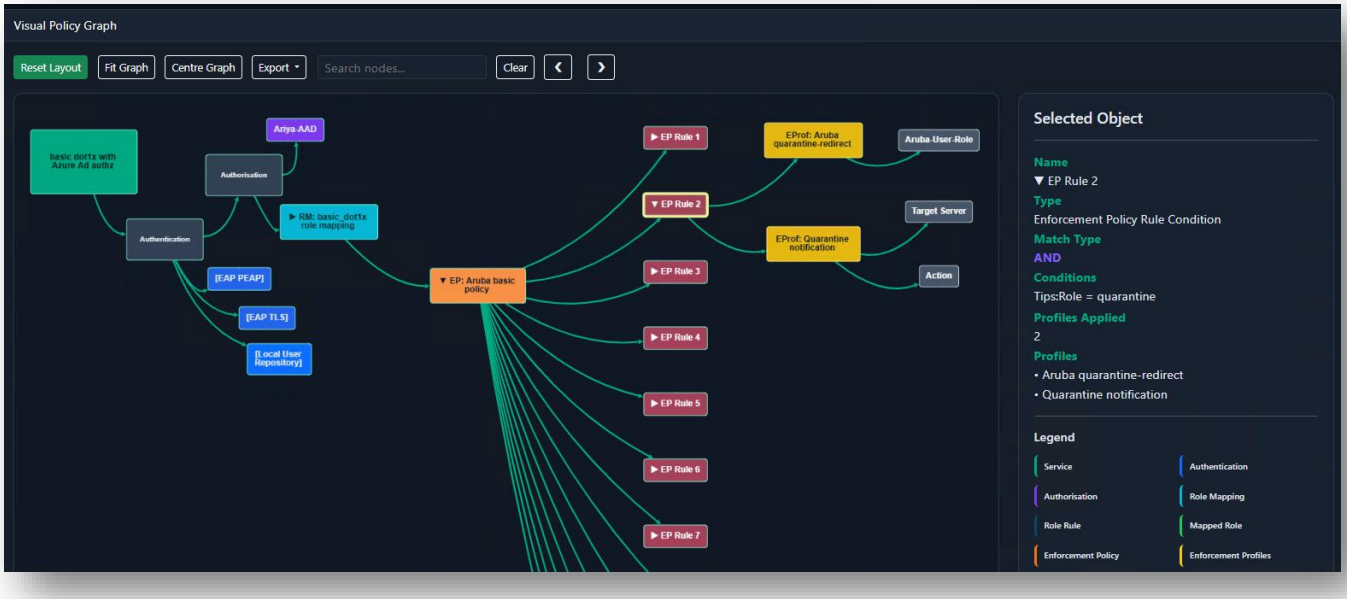
- Expandable and collapsible policy branches
- Automatic graph layout and node positioning
- Dynamic node sizing based on object names
- Zoom, fit-to-screen and centring controls
- Search with previous and next result navigation
- Contextual details for the selected object
- Direct Impact Analysis links for supported policy objects
- PNG, JPG and SVG export
- Visual highlighting when hovering over nodes and relationships

RM (Role Mapping) Policies and EP (Enforcement Policies) can be expanded progressively, allowing you to inspect only the level of detail you need.

In the example below, I've selected and expanded the basic_dot1x role mapping policy. The graph displays its mapping rules, the Roles assigned by those rules, and the configured default Role. The Selected Object panel shows additional information about the policy, including its evaluation algorithm, default Role, rule count, usage summary and Impact Analysis link.



You can expand the Enforcement Policy in the same way. In the example below, I've expanded Aruba basic policy to display its policy rules, conditions, Enforcement Profiles and Enforcement Attributes. Selecting a rule or profile updates the Selected Object panel with the corresponding condition, match type, applied profiles, attributes and other available details.



Complex Services may contain many rules, conditions, profiles and attributes. Progressive branch expansion keeps the graph readable by allowing you to start with the high-level workflow and reveal additional detail only when required. This makes it easier to investigate how authentication decisions are made without displaying the entire policy structure at once.

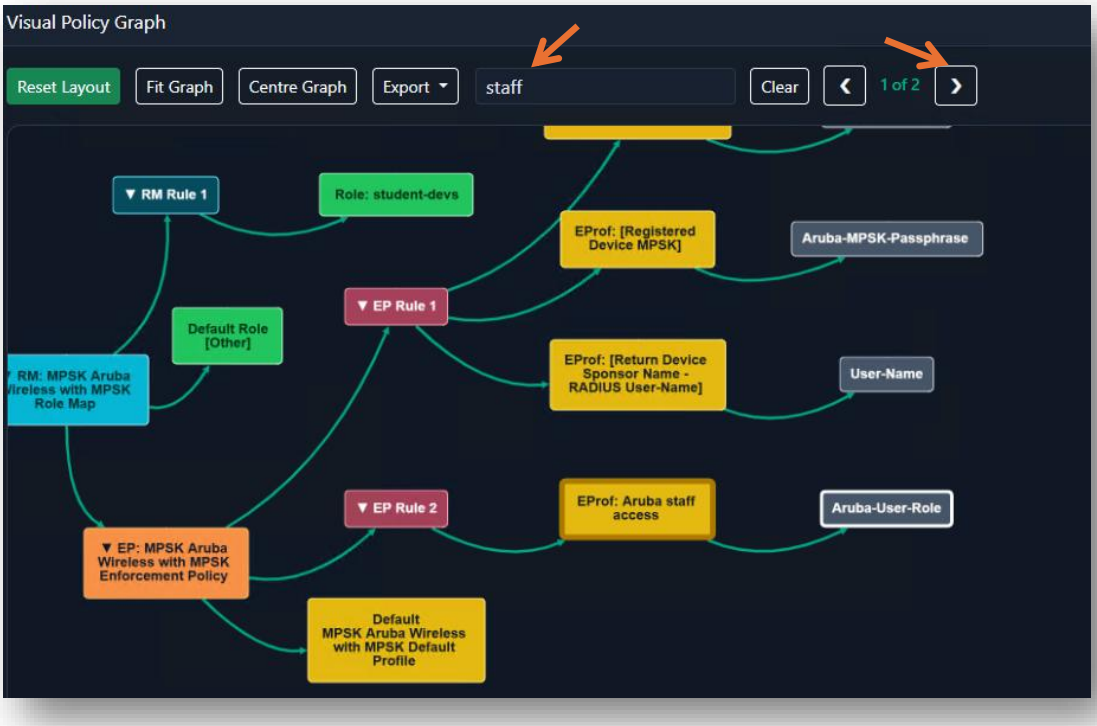
5.3 Advanced Search

The Visual Policy Graph includes an advanced search function that helps you locate objects and values within the current Service graph. You can search for information such as:

- Role names
- Enforcement profile names
- Enforcement policy names
- Authentication & Authorisation sources
- Attribute names and values
- Default roles and profiles
- Conditions and actions

As you enter a search term, the Visualiser automatically expands any collapsed branches containing a match. Matching nodes are highlighted, and the result counter shows the current result and the total number of matches.

In the example below, I've searched for staff. The Visualiser has found two matching nodes and highlighted the first result. You can use the previous and next controls beside the search field to move between the matches.



Search functionality includes:

- Automatic expansion of matching policy branches
- Highlighting of matching nodes and related paths
- Result counting
- Previous and next result navigation
- Active-result tracking
- A Clear option for resetting the search

This is particularly useful when working with a large Service graph containing many Role Mapping rules, Enforcement Policy rules, Roles, profiles and attributes.

The graph can also be exported for documentation, design reviews or operational records. From the Export menu, you can download the current visible graph in PNG, JPG or SVG format.



The exported graph reflects the current visible state, including the branches you have expanded. Collapsed branches remain excluded, which allows you to export only the level of detail required for the document or review.

5.4 Impact Analysis

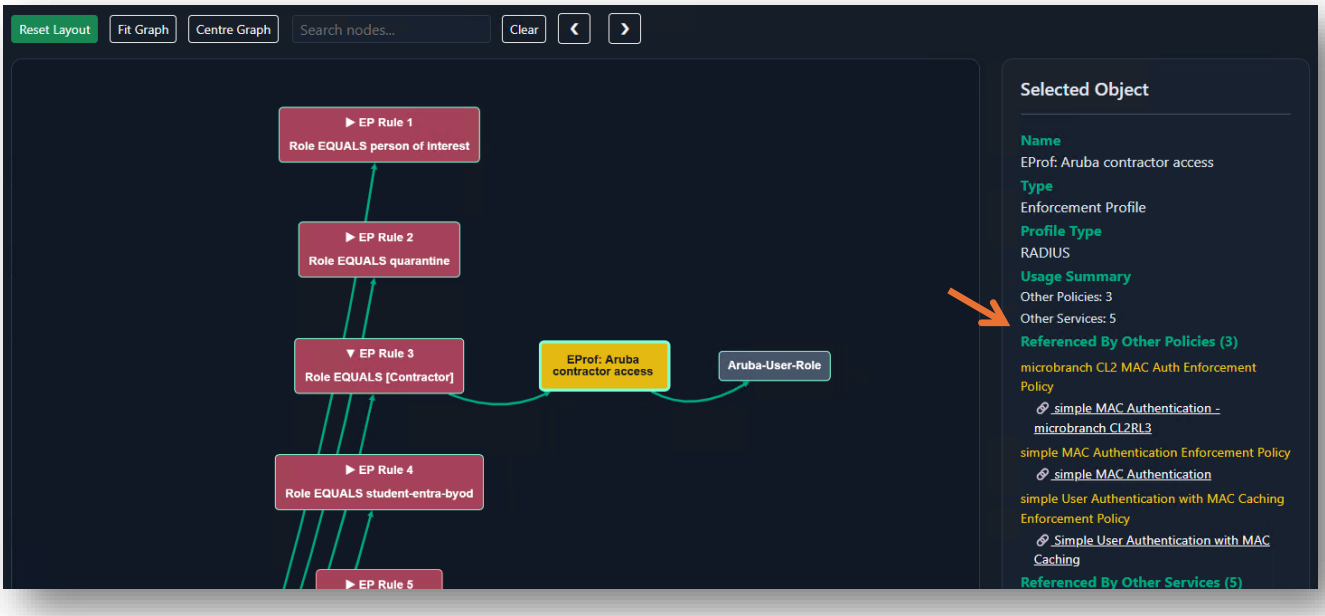
One of the most useful features in CPV is the ability to see where a policy object is referenced and understand what may be affected if that object is changed. You can access Impact Analysis directly from supported nodes in the Visual Policy Graph. When you select an Enforcement Profile, Enforcement Policy or Role Mapping Policy, the Selected Object panel displays the information available for that object, together with a View Impact Analysis link.

In the example below, I've selected an Enforcement Profile. The Selected Object panel shows the profile type, usage summary, action, attributes, other referencing policies and other affected Services.



The 🔗 links displayed beside referenced Services allow you to open the corresponding Service graph. The View Impact Analysis link opens the full report in a new browser tab, which preserves the current graph layout, expanded branches and selected object.

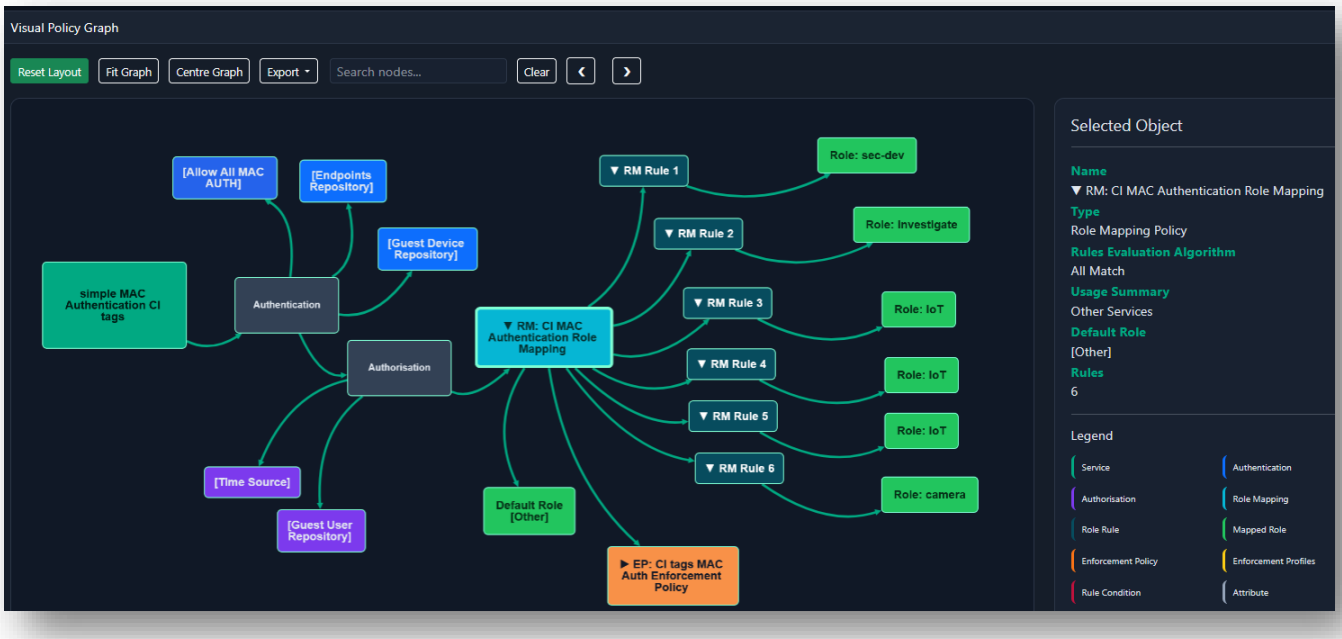
CPV also identifies when an Enforcement Profile is shared by other Enforcement Policies. In the example below, the selected profile is referenced by three policies. Where those policies are assigned to Services, the associated Services are also displayed.



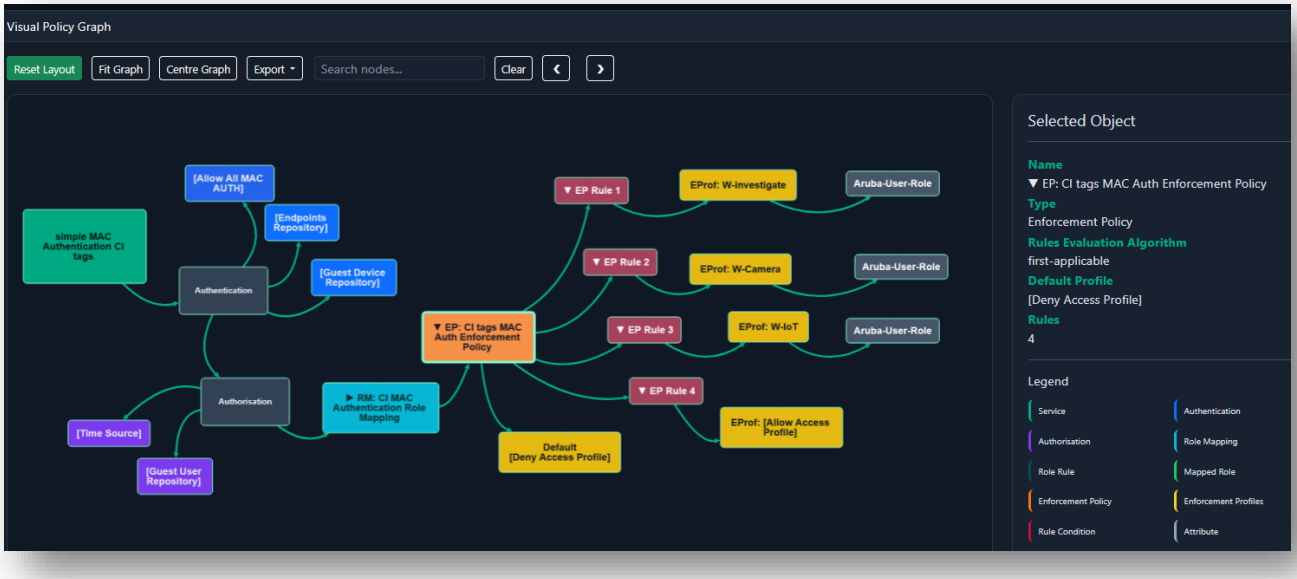
This information helps you understand whether a profile is isolated to one workflow or shared across multiple policies and Services before you modify it. Impact Analysis is available for the following ClearPass objects:

- Enforcement Profiles
- Enforcement Policies
- Role Mapping Policies and Roles

For a Role Mapping Policy, the graph shows the mapping rules, the Roles assigned by those rules and the configured default Role. Selecting the Role Mapping Policy also provides direct access to its Impact Analysis report.



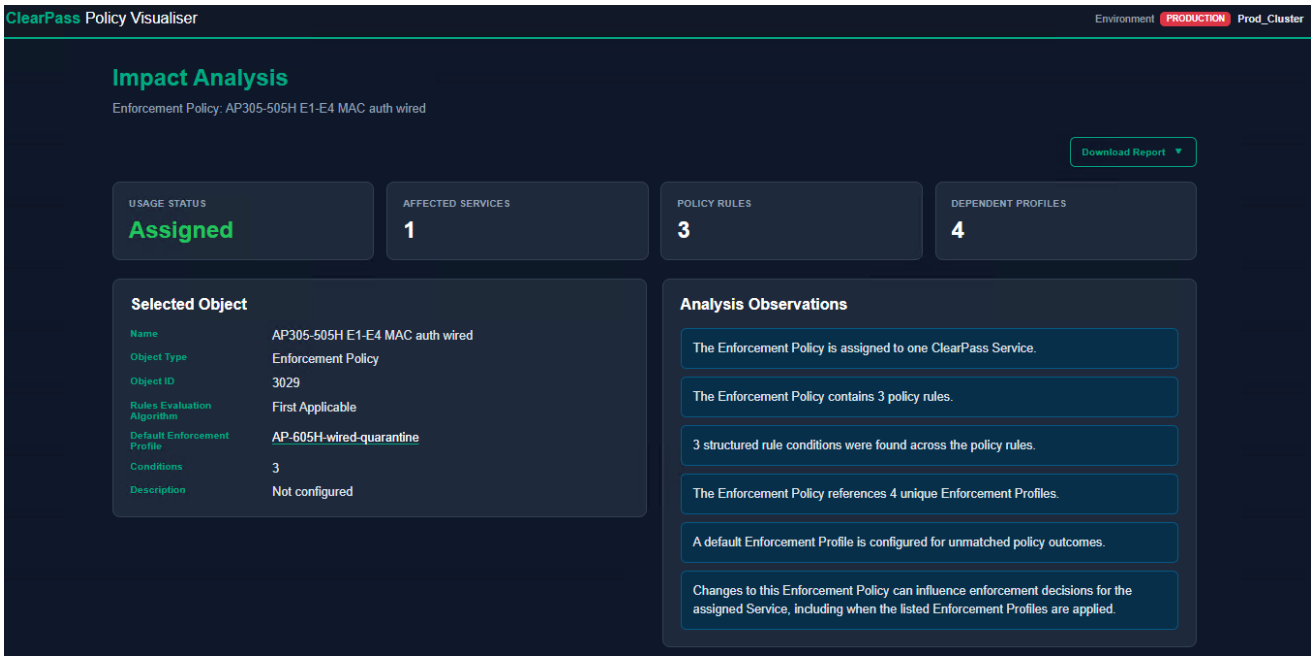
For an Enforcement Policy, the graph shows the policy rules, associated Enforcement Profiles, Enforcement Attributes and the configured default Enforcement Profile. Selecting the Enforcement Policy provides direct access to the corresponding Impact Analysis report.



Enforcement Policy Impact Analysis

The detailed Impact Analysis report provides a structured view of the selected object and its dependencies. In the example below, the summary shows that the Enforcement Policy is assigned to one Service, contains three policy rules and references four dependent Enforcement Profiles.

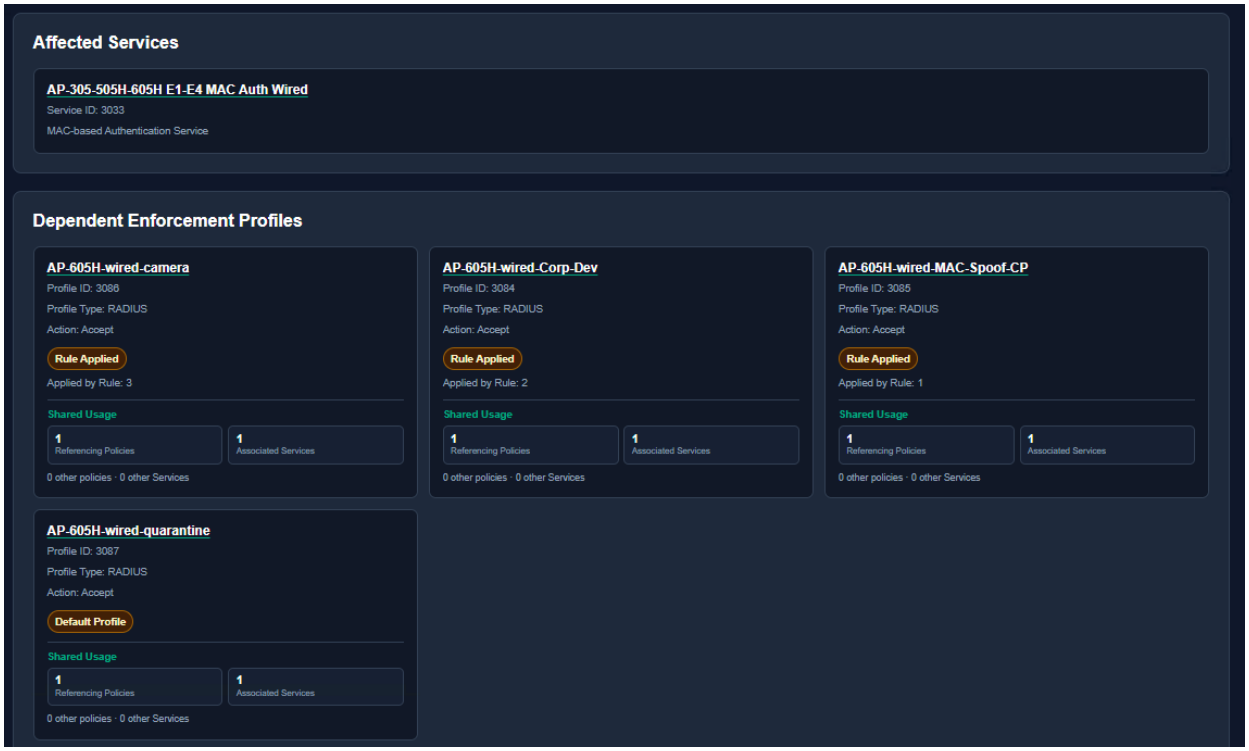
The Selected Object section displays the policy ID, evaluation algorithm, default Enforcement Profile, condition count and description. The Analysis Observations section summarises the discovered policy scope and highlights how changes may influence enforcement decisions for the assigned Service.



Further down the report, the Affected Services section identifies every discovered Service that uses the selected Enforcement Policy. You can select a Service name to open its dependency graph. The Dependent Enforcement Profiles section lists each unique Enforcement Profile referenced by the policy. Each profile card shows available information such as:

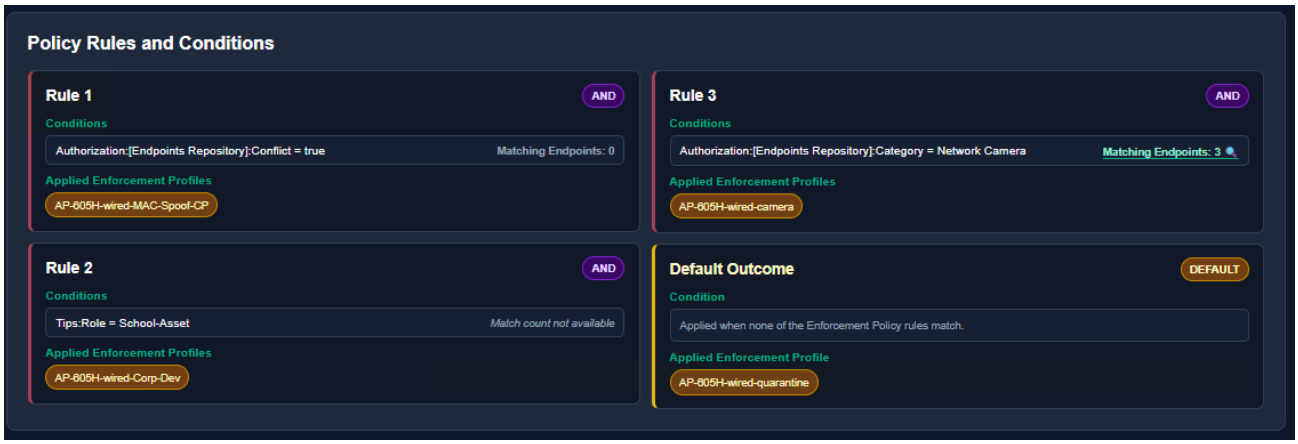
- Profile ID and Type
- Enforcement action
- Whether the profile is rule-applied or configured as the default
- The rule numbers that apply the profile
- Total referencing policies and Total associated Services
- Usage outside the selected policy and Service

A profile used by several rules is shown once in this inventory, with all applicable rule numbers retained. You can also select a profile name to open its Enforcement Profile Impact Analysis report.



The Policy Rules and Conditions section presents the processing logic in rule order. Each rule card displays its match type, structured conditions and applied Enforcement Profiles. Where a supported repository condition has matching results, the count is displayed as a link to Repository Search. A positive result includes a search icon and can be opened to inspect the matching endpoints. A confirmed count of zero is shown without a link, because there are no matching

endpoints to inspect. Where a match count cannot be determined from the configured data source, the report displays that the count is not available. The final Default Outcome card shows the Enforcement Profile applied when none of the policy rules match.



5.5 Endpoint Matching

Another useful feature of CPV is endpoint matching. Where a supported Role Mapping or Enforcement Policy condition references endpoint data, the Visualiser can display how many endpoints currently match that condition.

In the example below, I've selected Rule 4 from a Role Mapping Policy. The Selected Object panel shows two conditions:

- Device Insight Tags = [Computing Systems], matching 307 endpoints
- Authorization:[Endpoints Repository]:OS Family = Raspberry Pi, matching 2 endpoints

Because the rule uses an AND match type, the Visualiser also evaluates the conditions together. In this example, one endpoint matches the complete rule.

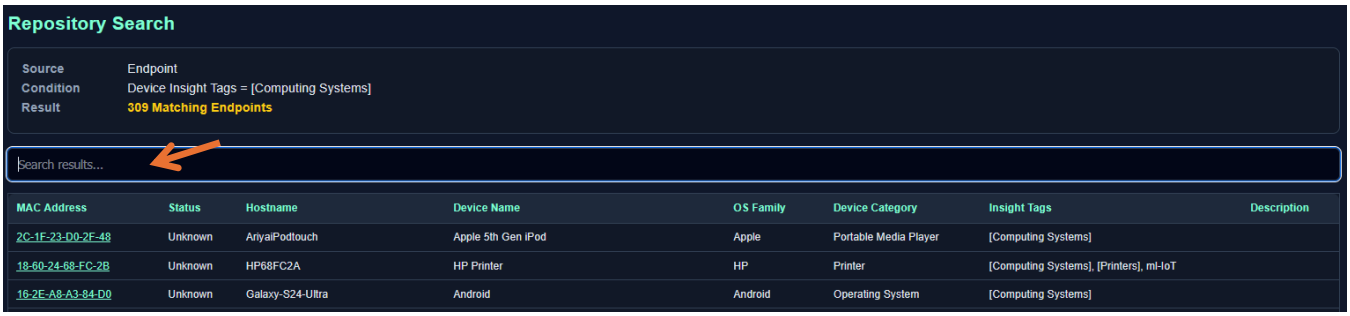


Where a condition has one or more matching endpoints, the match count includes a search icon. Select the link to open Repository Search for that specific condition.

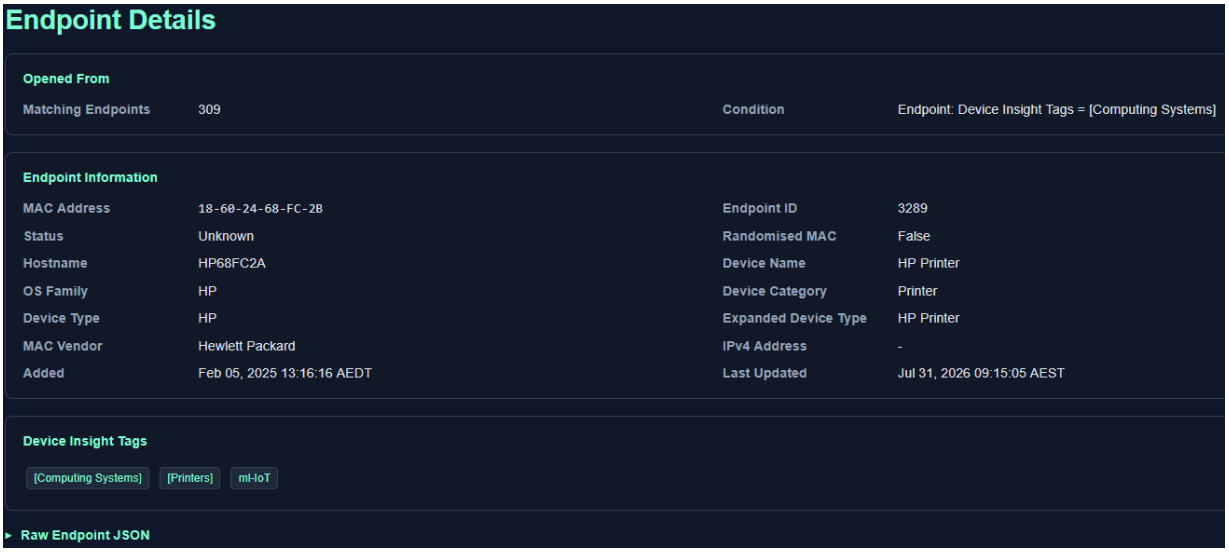
The Repository Search page lists the matching endpoints and displays available information such as:

- MAC address
- Status
- Hostname
- Device name
- Operating system family
- Device category
- Insight Tags
- Description

You can also use the search field to filter the returned results.



Select an endpoint to open the Endpoint Details page. This page displays the available endpoint and profiling attributes, including the MAC address, hostname, device category, device type, vendor, IP address and Insight Tags. The raw endpoint JSON is also available when deeper inspection is required.

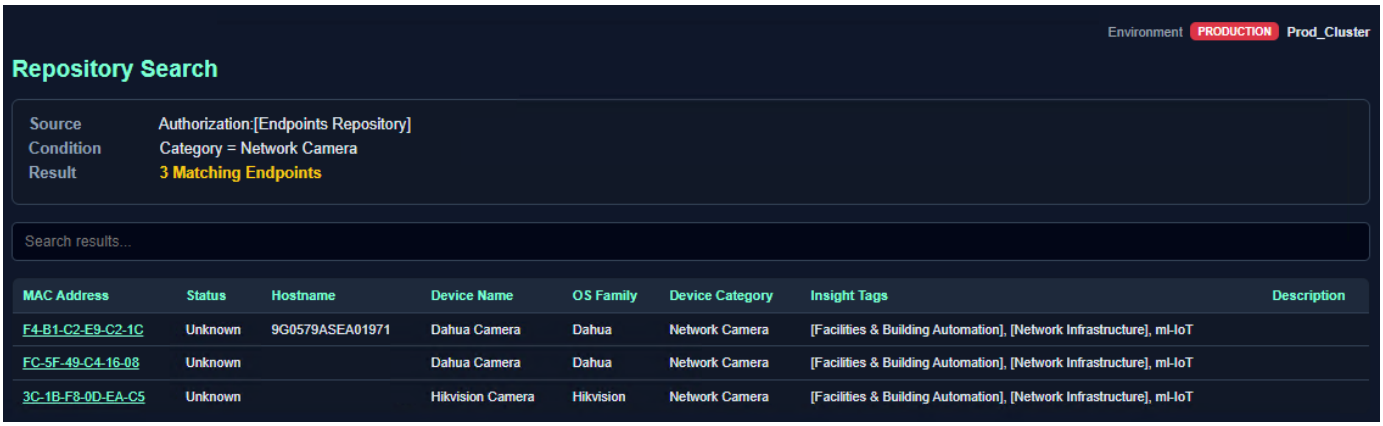


Endpoint matching is also available for supported conditions in Enforcement Policies. In the example below, Rule 3 contains the condition: Authorization:[Endpoints Repository]:Category = Network Camera

The condition currently matches three endpoints, as shown in the Selected Object panel.



Selecting the match-count link opens Repository Search and displays the three matching camera endpoints.



The Visualiser distinguishes between the following endpoint-count states:

- A positive count is displayed as a link to Repository Search.
- A confirmed count of zero is displayed without a link because there are no matching endpoints to inspect.
- Match count not available, indicates that a count could not be determined from the configured repository or profiling data source.

For rules containing multiple conditions, the individual condition counts show how many endpoints match each condition. Where available, the Rule Match Summary shows how many endpoints satisfy the complete rule after its AND or OR logic is applied. This makes endpoint matching useful when validating policy conditions, investigating unexpected authentication outcomes, and confirming whether a proposed rule is likely to affect existing endpoints.

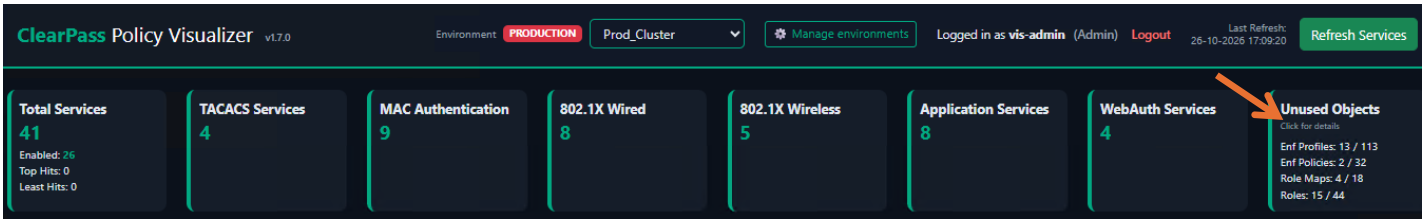
5.6 Unused Objects

The Unused Objects capability helps you identify ClearPass configuration objects that may no longer be required. Rather than relying on naming conventions, CPV analyses relationships across Services, policy rules, assigned and default Roles, Enforcement Policies, Enforcement Profiles and built-in mappings. The results provide practical candidates for configuration review and cleanup.

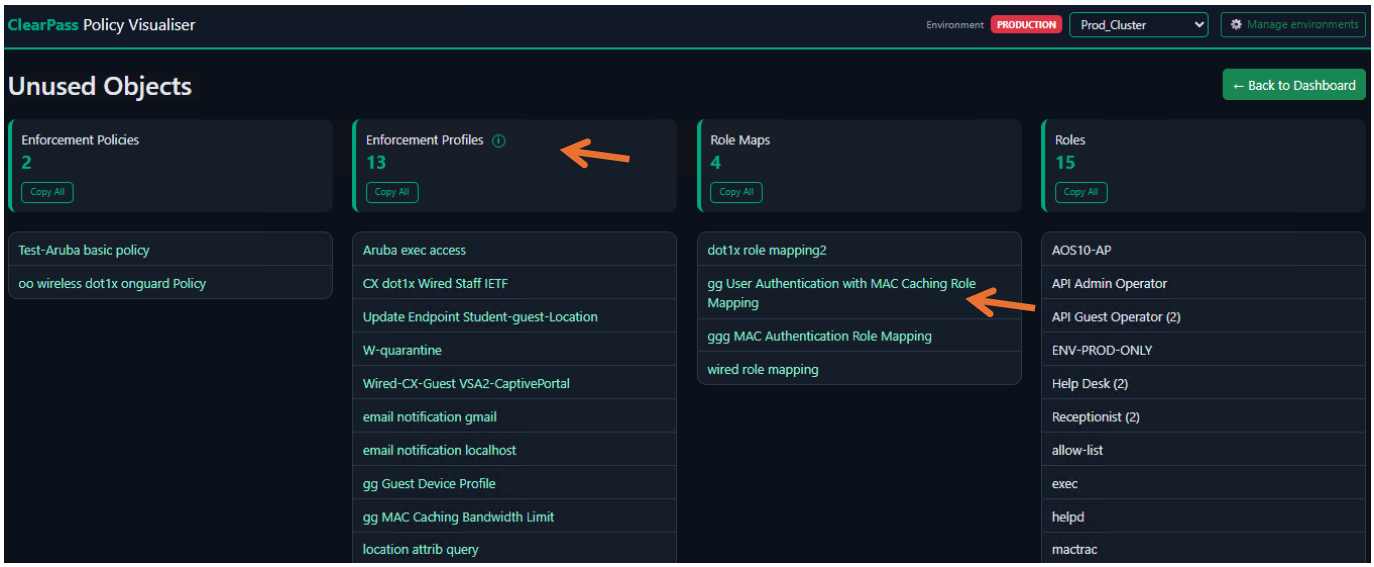
The Visualiser can identify unused:

- Enforcement Policies
- Enforcement Profiles
- Role Mapping Policies
- Roles

The Unused Objects card on the dashboard provides a summary for each object type. The figures show the number of unused objects compared with the total number discovered in that category. Select the card to open the detailed Unused Objects page.



The detailed page groups the results by object type and displays the unused count for each category. You can use Copy All to copy the complete list from a category for review or documentation. The results are intended as review candidates. CPV does not delete or modify any ClearPass configuration.

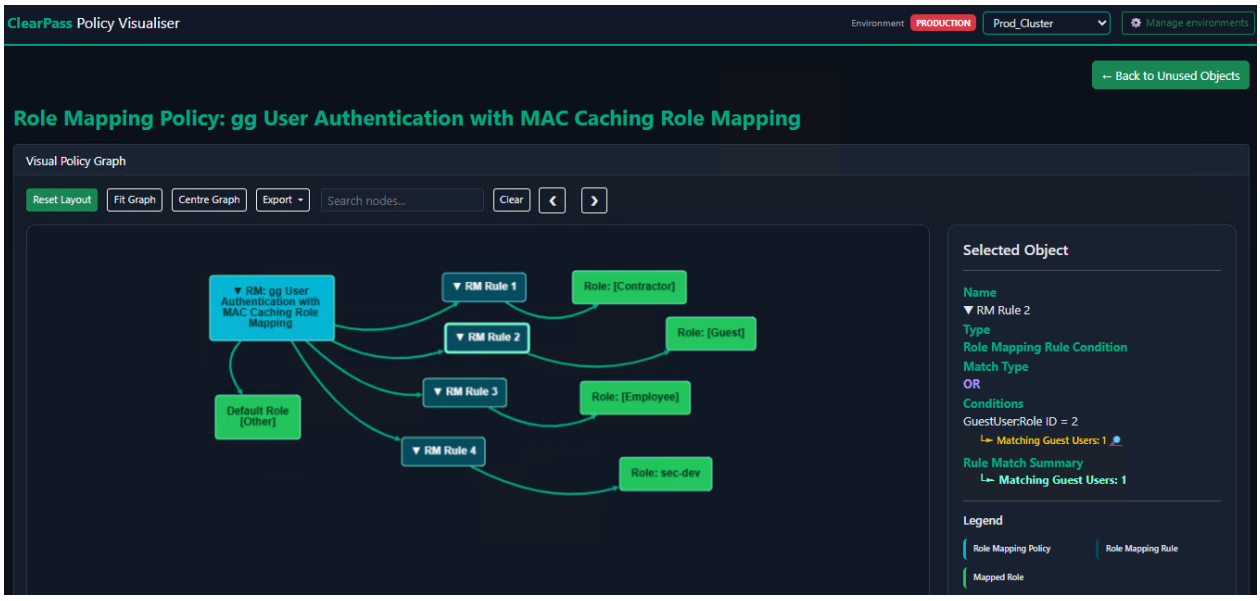


There is an important distinction for Enforcement Profiles. A profile is counted as unused only when no Enforcement Policy references it. If a profile is referenced by an Enforcement Policy that is not currently assigned to a Service, the profile is excluded from the unused count because a policy dependency still exists. The information icon beside Enforcement Profiles explains this classification.

You can also inspect supported unused objects directly from this page:

- Select an unused Enforcement Policy to open its standalone policy graph.
- Select an unused Role Mapping Policy to open its standalone policy graph.
- Select an unused Enforcement Profile to review its profile details.
- Unused Roles are listed for review and can be copied with the category list.

In the example below, I've selected the gg User Authentication with MAC Caching Role Mapping policy. The standalone graph shows the Role Mapping rules, mapped Roles and configured default Role without requiring the policy to be associated with a Service.



You can progressively expand the policy branches, select individual nodes and review the available information in the Selected Object panel. The standalone graph also supports search, layout controls and graph export.

For unused Enforcement Policies and Role Mapping Policies, the View Impact Analysis link provides an additional verification step. The report confirms that the policy is not assigned to a discovered Service while still showing its rules, conditions, dependent Enforcement Profiles or mapped Roles, and configured default outcome.

By combining unused-object detection with standalone graphs and Impact Analysis, the Visualiser helps reduce configuration clutter while ensuring that each reported object is reviewed in context rather than removed automatically.

6 Appendix A – Large Scale ClearPass Deployment Validation

ClearPass Policy Visualiser v1.6.1 was validated against a substantially larger ClearPass deployment to assess its behaviour with a high endpoint count. The deployment is one of the largest CPPM deployments in SouthPac.

During startup, CPV successfully loaded **1,375,838 endpoint fingerprint entries** directly from the ClearPass PostgreSQL database. The endpoint profiling cache was populated in **557.652 seconds**, approximately **9 minutes 18 seconds**.

The environment contained **53 Services**, **95 Roles**, **22 Role Mapping Policies**, and **47 Enforcement Policies**. The Enforcement Profile reference cache contained **232 profiles**, while the Impact Analysis lookup cache contained **435 policy objects**.

The Enforcement Profile reference cache was built in **243.374 seconds**, and the Role reference cache identified **92 referenced Roles** in **171.226 seconds**.

The complete CPV cache initialisation sequence finished successfully approximately **17 minutes 42 seconds** after startup. This included Service discovery, Insight Service Hit Counts, endpoint profiling, Role and policy reference caches, Unused Objects analysis and the Impact Analysis lookup cache.

The CPV host was connected to the ClearPass deployment **over a VPN during this test**. The measured times should therefore be treated as observed results from this particular test rather than as a performance benchmark. Network latency and VPN characteristics may have contributed to the overall processing time, particularly for operations requiring repeated interaction with ClearPass.

Cache initialisation performance can vary between environments depending on factors including the size of the ClearPass dataset, policy configuration, database performance, available resources and network path. The results demonstrate that CPV successfully completed its full cache initialisation against a ClearPass environment containing more than **1.375 million endpoint fingerprint entries.**

6.1 Test Summary

1,375,838 Endpoint fingerprint entries	**232** Enforcement Profiles in the reference cache
53 Services	**435** Impact Analysis lookup objects
95 Roles	
22 Role Mapping Policies	**~9 min 18 sec** endpoint profiling cache load
47 Enforcement Policies	**~17 min 42 sec** total cache initialisation

6.2 Docker Console Log

```
2026-09-30 00:36:43.800 | 30-09-2026 00:36:43 INFO Starting ClearPass Policy Visualiser
after initial setup...
2026-09-30 00:36:43.800 | 30-09-2026 00:36:43 INFO
============================================================
2026-09-30 00:36:43.801 | 30-09-2026 00:36:43 INFO ClearPass Policy Visualiser v1.6.1
Starting
2026-09-30 00:36:43.801 | 30-09-2026 00:36:43 INFO
============================================================
2026-09-30 00:36:43.801 | 30-09-2026 00:36:43 INFO Loading ClearPass services...
2026-09-30 00:36:49.557 | 30-09-2026 00:36:49 INFO Loading current-hour Service hit
counts from ClearPass Insight...
2026-09-30 00:36:51.955 | 30-09-2026 00:36:51 INFO Current-hour Service hit counts
loaded: 0 Services with hits.
2026-09-30 00:36:51.956 | 30-09-2026 00:36:51 INFO Service Hit Counts applied: 0
Services with current-hour hits.
2026-09-30 00:36:51.956 | 30-09-2026 00:36:51 INFO Services cached: 53
2026-09-30 00:36:51.956 | 30-09-2026 00:36:51 INFO Running initial health check...
```

```
2026-09-30 00:36:51.956 | 30-09-2026 00:36:51 INFO Running health check...
2026-09-30 00:36:57.363 | 30-09-2026 00:36:57 INFO Pre-loading endpoint data...
2026-09-30 00:36:57.402 | 30-09-2026 00:36:57 INFO Pre-loading endpoint profiling data
using PostgreSQL...
2026-09-30 00:46:14.313 | 30-09-2026 00:46:14 INFO Endpoint profiling SQL load complete:
1375838 rows in 556.911s
2026-09-30 00:46:15.010 | 30-09-2026 00:46:15 INFO Endpoint pre-load complete using
PostgreSQL: 1375838 fingerprint entries in 557.652s
2026-09-30 00:46:15.010 | 30-09-2026 00:46:15 INFO Building Role Cache...
2026-09-30 00:46:20.693 | 30-09-2026 00:46:20 INFO Roles cached: 95
2026-09-30 00:46:20.693 | 30-09-2026 00:46:20 INFO Building Enforcement Profile
Reference Cache...
2026-09-30 00:46:33.177 | 30-09-2026 00:46:33 INFO Retrieved 47 Enforcement Policies
from ClearPass
2026-09-30 00:50:24.045 | 30-09-2026 00:50:24 INFO Enforcement profile reference cache
built: 232 profiles, 47 total policies, 36 Service-assigned policies in 243.374s
2026-09-30 00:50:24.053 | 30-09-2026 00:50:24 INFO Building Role Mapping Reference
Cache...
2026-09-30 00:50:29.205 | 30-09-2026 00:50:29 INFO Building Role Reference Cache...
2026-09-30 00:53:10.964 | 30-09-2026 00:53:10 INFO Retrieved 47 Enforcement Policies
from ClearPass
2026-09-30 00:53:20.412 | 30-09-2026 00:53:20 INFO Role reference cache built: 92
referenced Roles in 171.226s
2026-09-30 00:53:20.412 | 30-09-2026 00:53:20 INFO Role references cached: 92
2026-09-30 00:53:20.412 | 30-09-2026 00:53:20 INFO Building Unused Objects Cache...
2026-09-30 00:54:06.228 | 30-09-2026 00:54:06 INFO Building Impact Analysis Lookup
Cache...
2026-09-30 00:54:15.967 | 30-09-2026 00:54:15 INFO Retrieved 263 Enforcement Profiles
from ClearPass
2026-09-30 00:54:20.897 | 30-09-2026 00:54:20 INFO Retrieved 47 Enforcement Policies
from ClearPass
2026-09-30 00:54:25.940 | 30-09-2026 00:54:25 INFO Impact Analysis lookup objects
cached: 435
2026-09-30 00:54:25.940 | 30-09-2026 00:54:25 INFO Cache initialisation complete.
2026-09-30 00:54:25.940 | 30-09-2026 00:54:25 INFO Services: 53
2026-09-30 00:54:25.940 | 30-09-2026 00:54:25 INFO Role Mapping Policies: 22
2026-09-30 00:54:25.940 | 30-09-2026 00:54:25 INFO Enforcement Profiles: 232
2026-09-30 00:54:25.940 | 30-09-2026 00:54:25 INFO Impact Analysis lookup objects: 435
2026-09-30 00:54:25.940 | 30-09-2026 00:54:25 INFO Last Refresh: 30-09-2026 00:36:43
2026-09-30 00:54:25.940 | 30-09-2026 00:54:25 INFO ClearPass Policy Visualiser started
successfully.
```

7 Appendix B – CLI based ClearPass Policy Visualiser Configuration recovery

Initial Setup creates and validates the CPV configuration during the first deployment. There may be occasions where you need to change an existing setting later, for example after changing a ClearPass server address or rotating a REST API Client Secret, RADIUS shared secret or PostgreSQL password.

From v1.7.0, the recommended way to do this is from the dashboard: select **Manage environments**, then **Edit** next to the environment. The new settings are tested before they are saved, the previous settings are kept as /data/.visualiser.env.prev, and the environment reloads without a restart. The manual procedure below is still useful when CPV cannot start, for example after a credential change on the ClearPass side.

For Docker deployments, the active Visualiser configuration is stored in /data/.visualiser.env within the persistent clearpass-visualiser-data volume. Before making any changes, I recommend creating a backup of the current configuration.

Before modifying the configuration, stop the Visualiser and make a copy of the existing .visualiser.env file. The configuration contains credentials and should be handled as sensitive data.

List the existing files.

```
C:\PythonScriptingPostman\clearpass-policy-visualiser>docker exec clearpass-policy-visualiser ls -
la /data
total 16
drwxr-xr-x 2 visualiser visualiser 4096 Sep 12 10:02 .
drwxr-xr-x 1 root       root       4096 Sep 13 13:49 ..
-rw------- 1 visualiser visualiser   64 Sep 11 13:26 .flask_secret
-rw------- 1 visualiser visualiser  725 Sep 12 10:01 .visualiser.env

C:\PythonScriptingPostman\clearpass-policy-visualiser>
```

Back up the existing configuration

```
C:\PythonScriptingPostman\clearpass-policy-visualiser>docker exec -u 0 clearpass-policy-visualiser
cp -p /data/.visualiser.env /data/.visualiser.env.bak

C:\scripts\ClearPass-visual>docker exec clearpass-policy-visualiser ls -la /data
total 20
drwxr-xr-x 2 visualiser visualiser 4096 Sep 13 13:51 .
drwxr-xr-x 1 root       root       4096 Sep 13 13:49 ..
-rw------- 1 visualiser visualiser   64 Sep 11 13:26 .flask_secret
-rw------- 1 visualiser visualiser  725 Sep 12 10:01 .visualiser.env
-rw------- 1 visualiser visualiser  725 Sep 12 10:01 .visualiser.env.bak

C:\PythonScriptingPostman\clearpass-policy-visualiser>
```

Copy the configuration to Windows current directory

```
C:\PythonScriptingPostman\clearpass-policy-visualiser>docker cp clearpass-policy-
visualiser:/data/.visualiser.env .visualiser.env.edit
Successfully copied 725B (transferred 2.56kB) to C:\scripts\ClearPass-pyclearpass\ClearPass-
visual\.visualiser.env.edit

C:\PythonScriptingPostman\clearpass-policy-visualiser>
C:\PythonScriptingPostman\clearpass-policy-visualiser>dir .vis*
 Volume in drive C has no label.
 Volume Serial Number is ACB8-58FF

 Directory of C:\scripts\ClearPass-visual>

09/12/2026  10:01 AM               725 .visualiser.env.edit
               2 File(s)          1,389 bytes
               0 Dir(s)  157,562,146,816 bytes free

C:\PythonScriptingPostman\clearpass-policy-visualiser>
```

Open .visualiser.env.edit using a text editor and modify only the settings that need to change. Because the file contains ClearPass connection credentials, treat it as sensitive configuration data.

Now I edited the .visualiser.env.edit file and added an extra “#” and then copying it back to docker.

```
C:\PythonScriptingPostman\clearpass-policy-visualiser>docker cp .visualiser.env.edit clearpass-policy-visualiser:/data/.visualiser.env
Successfully copied 727B (transferred 2.56kB) to clearpass-policy-visualiser:/data/.visualiser.env

C:\PythonScriptingPostman\clearpass-policy-visualiser>docker exec clearpass-policy-visualiser ls -la /data
total 20
drwxr-xr-x 2 visualiser visualiser 4096 Sep 13 14:01 .
drwxr-xr-x 1 root       root       4096 Sep 13 14:01 ..
-rw------- 1 visualiser visualiser   64 Sep 11 13:26 .flask_secret
-rwxr-xr-x 1 root       root        727 Sep 13 14:01 .visualiser.env
-rw------- 1 visualiser visualiser  725 Sep 12 10:01 .visualiser.env.bak

C:\PythonScriptingPostman\clearpass-policy-visualiser>
```

After copying the configuration back into the container, restore the Visualiser ownership and restrictive file permissions before restarting the application.

```
C:\PythonScriptingPostman\clearpass-policy-visualiser>docker exec -u 0 clearpass-policy-visualiser chown visualiser:visualiser /data/.visualiser.env

C:\PythonScriptingPostman\clearpass-policy-visualiser>docker exec -u 0 clearpass-policy-visualiser chmod 600 /data/.visualiser.env

C:\PythonScriptingPostman\clearpass-policy-visualiser>docker exec clearpass-policy-visualiser ls -la /data
total 20
drwxr-xr-x 2 visualiser visualiser 4096 Sep 13 14:01 .
drwxr-xr-x 1 root       root       4096 Sep 13 14:01 ..
-rw------- 1 visualiser visualiser   64 Sep 11 13:26 .flask_secret
-rw------- 1 visualiser visualiser  727 Sep 13 14:01 .visualiser.env
-rw------- 1 visualiser visualiser  725 Sep 12 10:01 .visualiser.env.bak

C:\PythonScriptingPostman\clearpass-policy-visualiser>
```

And finally delete the temporary file we copied in windows directory.

```
C:\PythonScriptingPostman\clearpass-policy-visualiser>del .visualiser.env.edit
C:\PythonScriptingPostman\clearpass-policy-visualiser>dir .visualiser.env.edit
 Volume in drive C has no label.
 Volume Serial Number is ACB8-58FF

 Directory of C:\PythonScriptingPostman\clearpass-policy-visualiser

File Not Found

C:\PythonScriptingPostman\clearpass-policy-visualiser>
```

Now we need to restart the Visualiser for the updated configuration to take effect. If Docker Compose is running in attached mode, press Ctrl+C to stop the Visualiser and then run:

```
docker compose up
```

Then watch the console during startup and confirm that ClearPass connectivity, PostgreSQL endpoint profiling and cache initialisation complete successfully.