Gaia API Workshop

Khalid Al-Shawwaf
Solution Architect, October 2021

Duration: 2 Hours

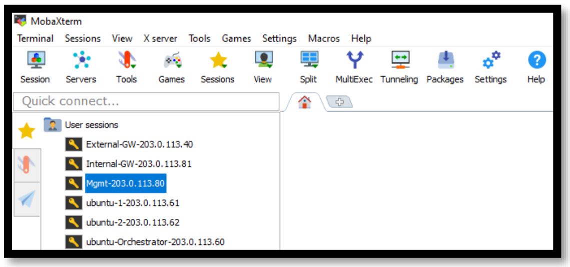
Introduction

In this workshop, we will learn how to use the Gaia API to manage the Gaia OS. There are multiple ways to use the API such as:

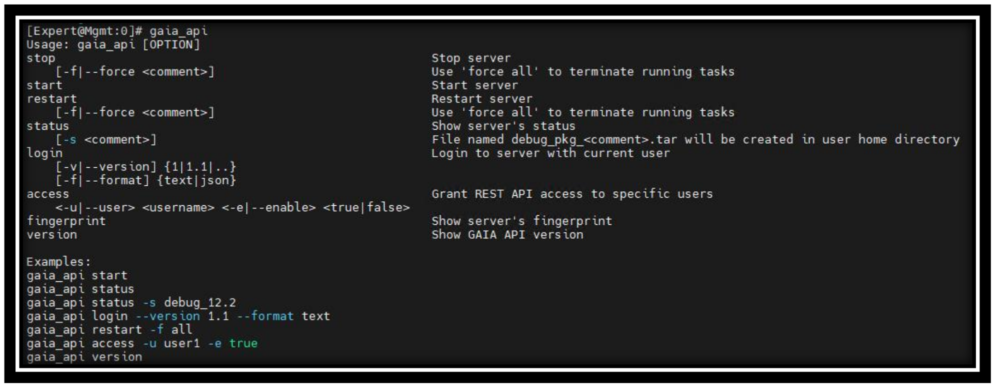
- Directly on the appliance using the mgmt_cli tool.
- Using Web tools such as Postman.
- Via the management API (Proxy).
- Using the Check Point Python SDK.

Exercise 1: Gaia API Infrastructure

1. Use the preinstalled SSH client MobaXterm to connect to the saved management server "Mgmt-203.0.113.80" (*double-click to connect*).

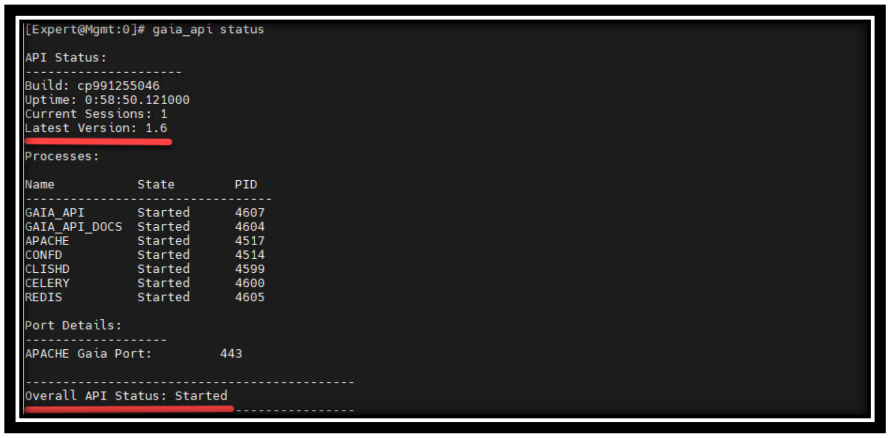


2. Run the command "gaia_api" to list all the available options.



```
[Expert@Mgmt:0]# gaia_api
Usage: gaia_api [OPTION]
stop                                                      Stop server
    [-f|--force <comment>]                                Use 'force all' to terminate running tasks
start                                                     Start server
restart                                                   Restart server
    [-f|--force <comment>]                                Use 'force all' to terminate running tasks
status                                                    Show server's status
    [-s <comment>]                                        File named debug_pkg_<comment>.tar will be created in user home directory
login                                                     Login to server with current user
    [-v|--version] {1|1.1|..}
    [-f|--format] {text|json}
access                                                    Grant REST API access to specific users
    <-u|--user> <username> <-e|--enable> <true|false>
fingerprint                                               Show server's fingerprint
version                                                   Show GAIA API version

Examples:
gaia_api start
gaia_api status
gaia_api status -s debug_12.2
gaia_api login --version 1.1 --format text
gaia_api restart -f all
gaia_api access -u user1 -e true
gaia_api version
```

3. Run the command "gaia_api status" and verify the status of the server.

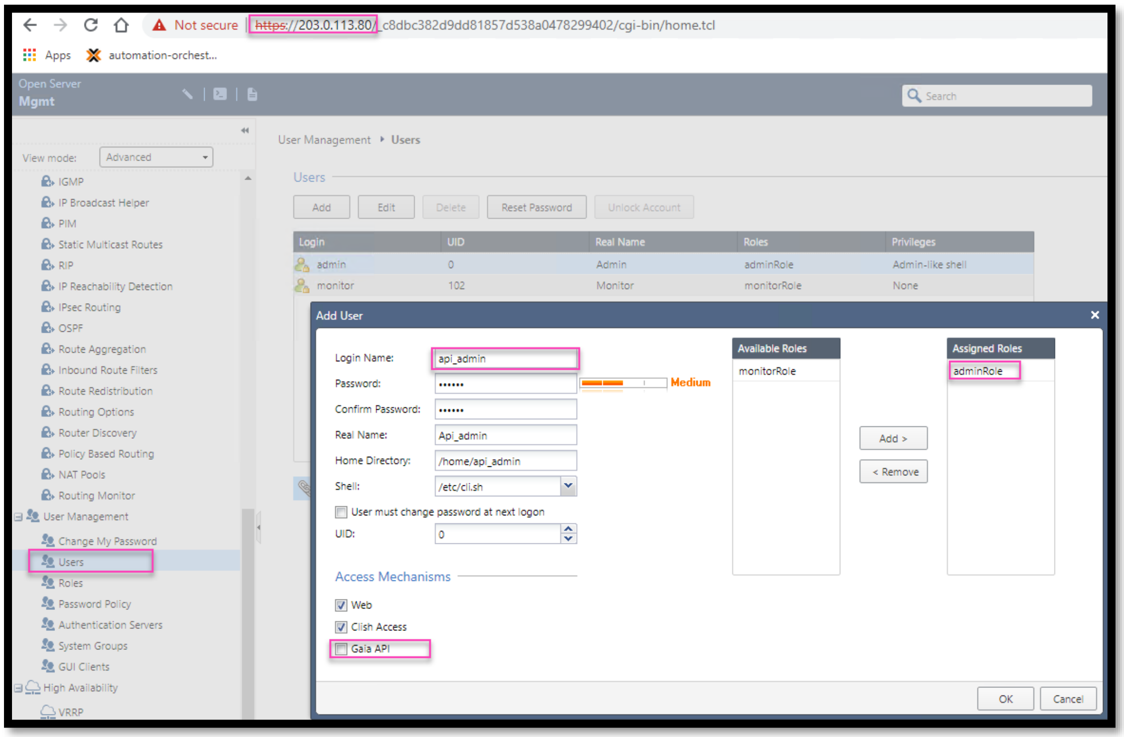


```
[Expert@Mgmt:0]# gaia_api status

API Status:
--------------------
Build: cp991255046
Uptime: 0:58:50.121000
Current Sessions: 1
Latest Version: 1.6

Processes:

Name           State        PID
-------------------------------
GAIA_API       Started      4607
GAIA_API_DOCS  Started      4604
APACHE         Started      4517
CONFD          Started      4514
CLISHD         Started      4599
CELERY         Started      4600
REDIS          Started      4605

Port Details:
------------------
APACHE Gaia Port:        443

---------------------------------------------
Overall API Status: Started
---------------------------------------------
```

- Note that the Gaia API has a login option. Access to the Gaia API requires a login via a user that is allowed access to the Gaia API infrastructure. Only predefined administrators are allowed by default.

4. Open a browser and log in to the Gaia Portal of the management server (https://203.0.113.80 username: admin, password: vpn123) and add a new user (Login name: *api_admin*, password: *vpn123*).

5. Note that the Gaia API access is unchecked by default, leave it **unchecked**.



Not secure | ~~https~~://203.0.113.80/_c8dbc382d9dd81857d538a0478299402/cgi-bin/home.tcl

User Management ▸ Users

Login	UID	Real Name	Roles	Privileges
admin	0	Admin	adminRole	Admin-like shell
monitor	102	Monitor	monitorRole	None

Add User

Login Name:	api_admin
Password:	•••••• (Medium)
Confirm Password:	••••••
Real Name:	Api_admin
Home Directory:	/home/api_admin
Shell:	/etc/cli.sh
UID:	0

☐ User must change password at next logon

Available Roles: monitorRole

Assigned Roles: adminRole

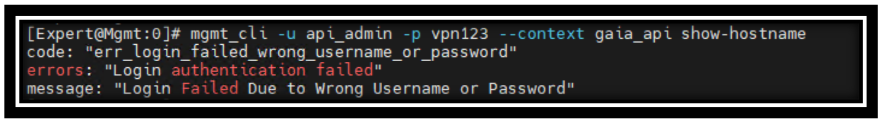
Access Mechanisms

☑ Web

☑ Clish Access

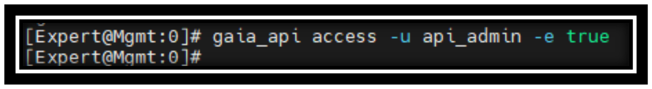
☐ Gaia API

6. Use the mgmt_cli tool to log in to the Gaia API server with the user we just created (api_admin/vpn123).



```
[Expert@Mgmt:0]# mgmt_cli -u api_admin -p vpn123 --context gaia_api show-hostname
code: "err_login_failed_wrong_username_or_password"
errors: "Login authentication failed"
message: "Login Failed Due to Wrong Username or Password"
```

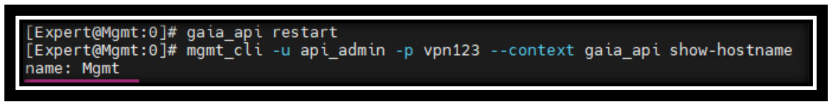
7. Use the command "*gaia_api access –u api_admin –e true*" to grant the user permission to log in to the Gaia API.



```
[Expert@Mgmt:0]# gaia_api access -u api_admin -e true
[Expert@Mgmt:0]#
```

- Note that for the permission to apply, we need to restart the Gaia API server using the command (gaia_api restart)

8. Run the mgmt_cli command from step 5 again and make sure you are getting the expected results.



```
[Expert@Mgmt:0]# gaia_api restart
[Expert@Mgmt:0]# mgmt_cli -u api_admin -p vpn123 --context gaia_api show-hostname
name: Mgmt
```

- Note: it is also possible to make the changes using the Gaia Portal via the user settings page or via the command (*add rba user api_admin access-mechanisms Gaia-API*)

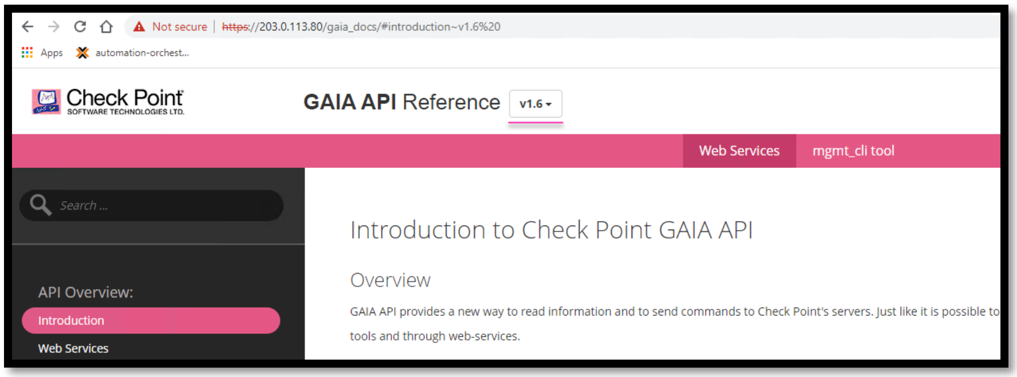
Exercise 2: Using Gaia API Documentation

The Gaia API documentation is available online and can be accessed via the link below:
https://sc1.checkpoint.com/documents/latest/GaiaAPIs/#introduction~v1.6%20

The documentation service is also a part of the Gaia API server. We will use the API reference to find the API commands for the mgmt_cli tool and also for the Web Services. All versions are accessible up to the latest version installed on the server.

1. Navigate to the Gaia documentations portal on the management server via the link (https://203.0.113.80/gaia_docs/).

- **Note** that the forward slash at the end is necessary.

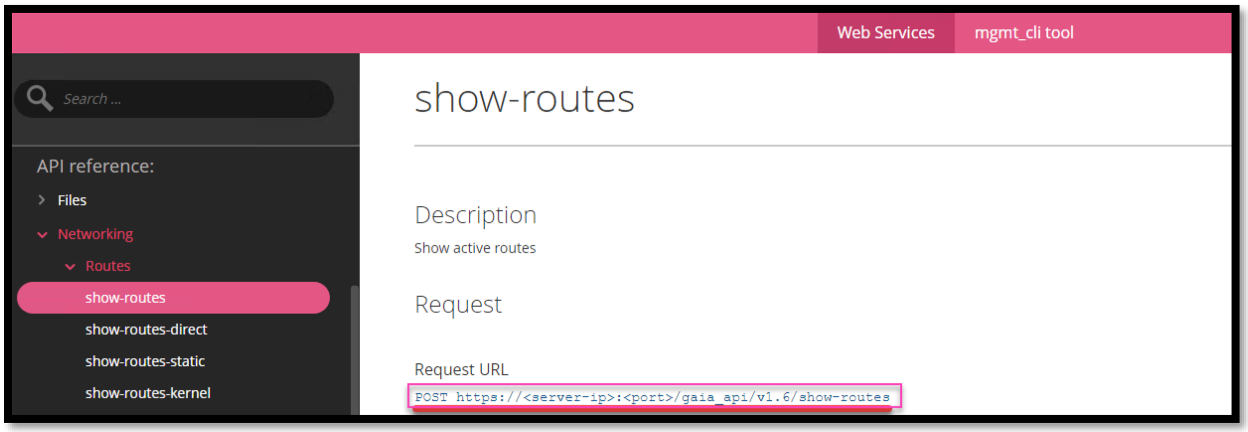


Introduction to Check Point GAIA API

Overview

GAIA API provides a new way to read information and to send commands to Check Point's servers. Just like it is possible to tools and through web-services.

2. From the menu on the left side, navigate to the routes section and select the show-routes command documentation (Networking -> Routes -> show-routes).



show-routes

Description

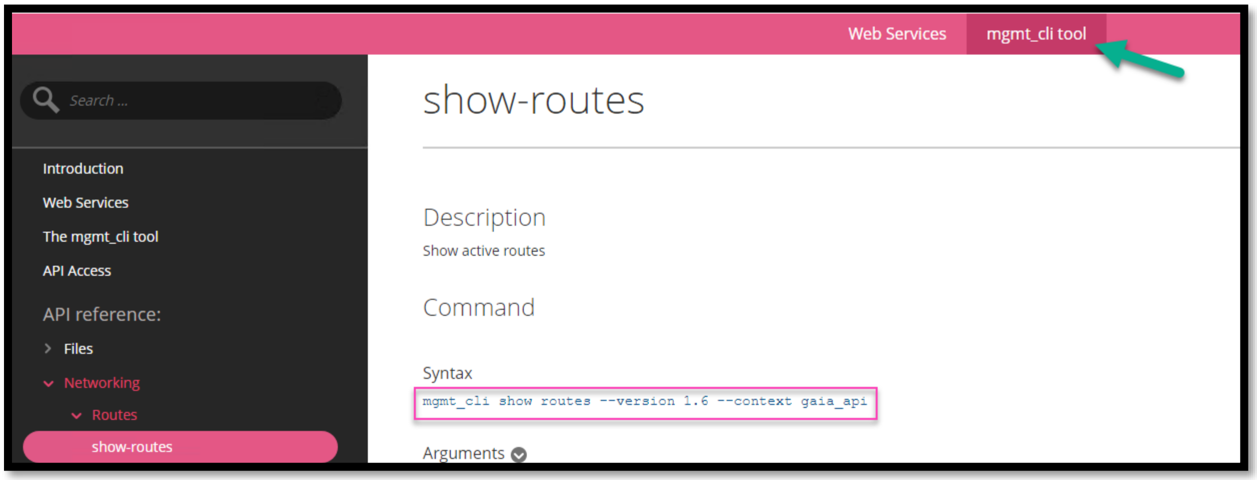
Show active routes

Request

Request URL

```
POST https://<server-ip>:<port>/gaia_api/v1.6/show-routes
```

3. The API reference defaults to the Web Service page, switch to the “mgmt_cli tool” section from the top menu, and check the command format when using mgmt_cli.



show-routes

Description

Show active routes

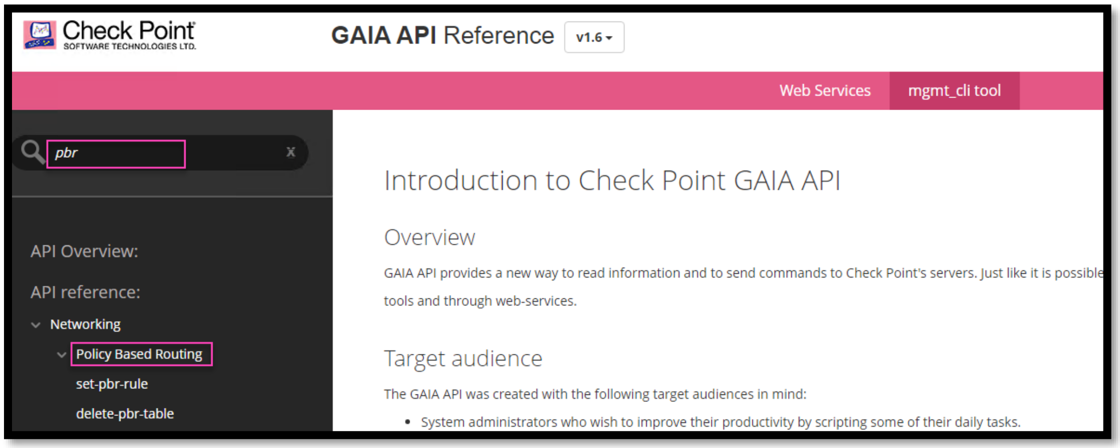
Command

Syntax

```
mgmt_cli show routes --version 1.6 --context gaia_api
```

Arguments

4. Use the search tab to search for policy-based routing (PBR) API calls.



Introduction to Check Point GAIA API

Overview

GAIA API provides a new way to read information and to send commands to Check Point's servers. Just like it is possible
tools and through web-services.

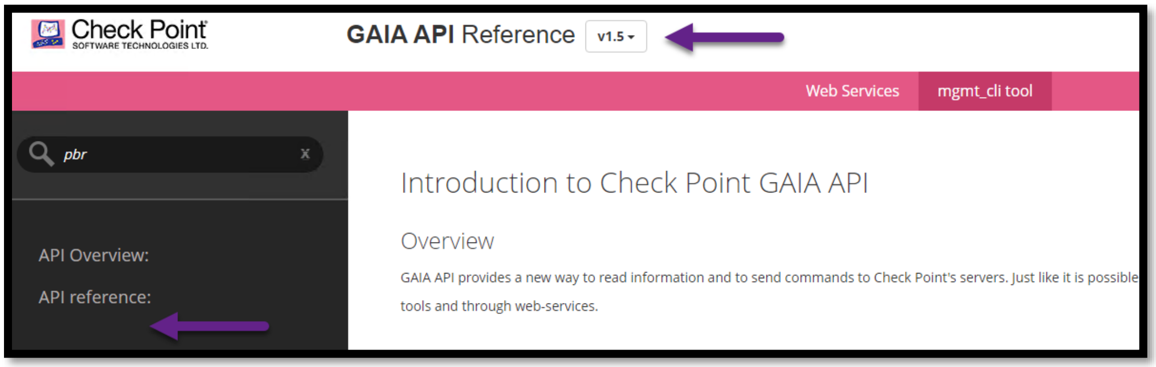
Target audience

The GAIA API was created with the following target audiences in mind:

- System administrators who wish to improve their productivity by scripting some of their daily tasks.

- Note that you need to search using the works expected in the API call. The search mechanism does not look for titles or section names. If you search for "Policy Based Routing", you will see no results.

5. Change the reference version from v1.6 to v1.5. Notice that the results disappeared.



Introduction to Check Point GAIA API

Overview

GAIA API provides a new way to read information and to send commands to Check Point's servers. Just like it is possible
tools and through web-services.

- Note: The policy-based routing was added in v1.6. Switching the reference version will only show documentation for v1.5, which does not include the Policy Based Routing API commands.
- The Changelogs for the Gaia API are documented under the Gaia API solution "sk143612".
- The version can be used in the API command using the argument (--version). For example:

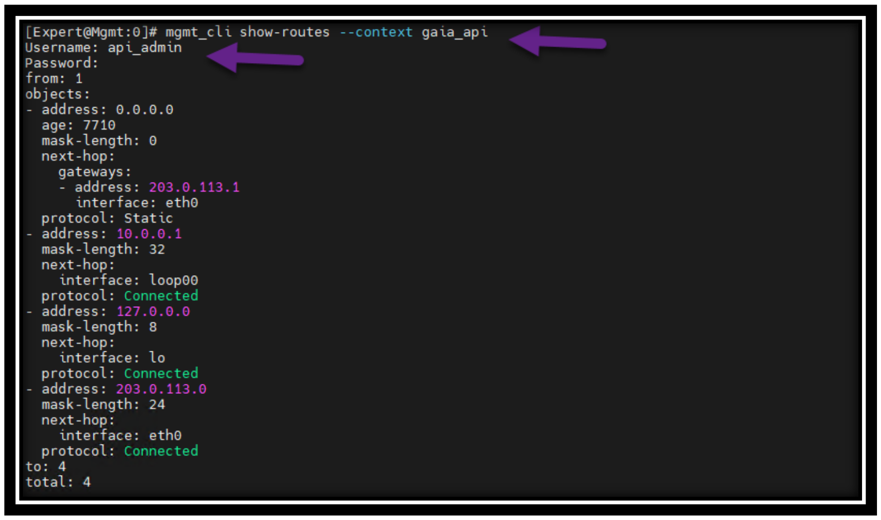
```
mgmt_cli show api-versions --version 1.5 --context gaia_api
```

6. Switch the version back to v1.6.

Exercise 3: Using mgmt_cli tool

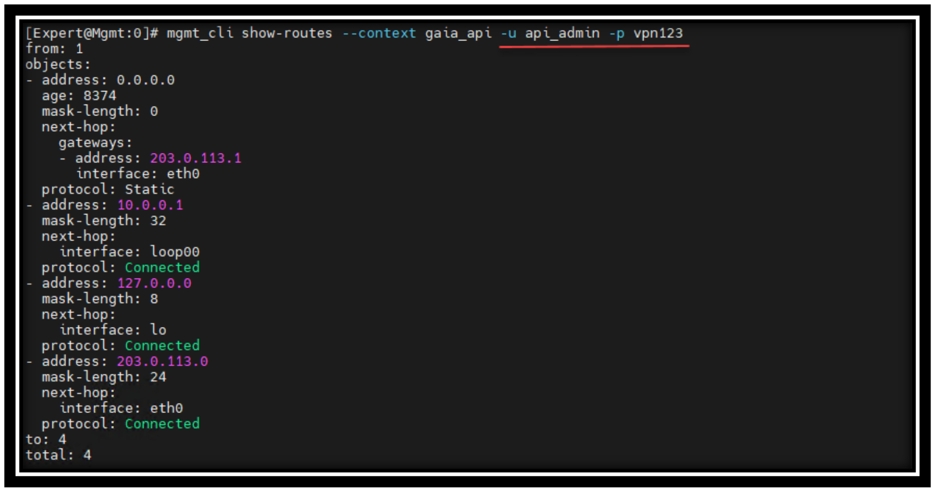
The mgmt_cli tool is installed as part of Gaia on all R80 gateways and can be used in scripts running in expert mode. It is also installed as part of the R80 SmartConsole installation (typically under C:\Program Files (x86)\CheckPoint\SmartConsole\R80\PROGRAM\) and can be copied to run on any Windows machine. In this exercise, we will use the tool to perform Gaia API calls.

1. Connect to the CLI of the management using Mobaxterm and issue the command (mgmt_cli show-routes --context gaia_api). Note that the mgmt_cli tool will prompt you for the credentials, as it was not provided with the command.



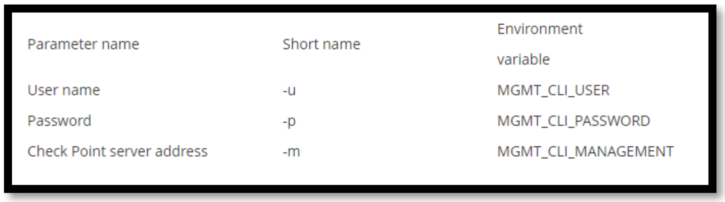
```
[Expert@Mgmt:0]# mgmt_cli show-routes --context gaia_api
Username: api_admin
Password:
from: 1
objects:
- address: 0.0.0.0
  age: 7710
  mask-length: 0
  next-hop:
    gateways:
    - address: 203.0.113.1
      interface: eth0
  protocol: Static
- address: 10.0.0.1
  mask-length: 32
  next-hop:
    interface: loop00
  protocol: Connected
- address: 127.0.0.0
  mask-length: 8
  next-hop:
    interface: lo
  protocol: Connected
- address: 203.0.113.0
  mask-length: 24
  next-hop:
    interface: eth0
  protocol: Connected
to: 4
total: 4
```

2. Run the command again and this time provide the credentials using the flag “-u” for the user name and “-p” for the password.



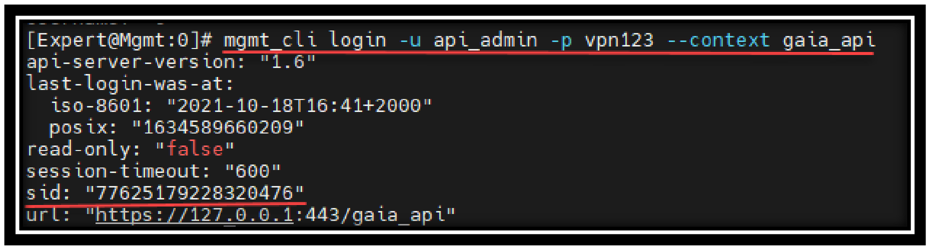
```
[Expert@Mgmt:0]# mgmt_cli show-routes --context gaia_api -u api_admin -p vpn123
from: 1
objects:
- address: 0.0.0.0
  age: 8374
  mask-length: 0
  next-hop:
    gateways:
    - address: 203.0.113.1
      interface: eth0
  protocol: Static
- address: 10.0.0.1
  mask-length: 32
  next-hop:
    interface: loop00
  protocol: Connected
- address: 127.0.0.0
  mask-length: 8
  next-hop:
    interface: lo
  protocol: Connected
- address: 203.0.113.0
  mask-length: 24
  next-hop:
    interface: eth0
  protocol: Connected
to: 4
total: 4
```

- Note that it is possible to use environment variables with the mgmt_cli tool. The table below shows the variables that can be exported instead of providing the user name and password on the terminal.



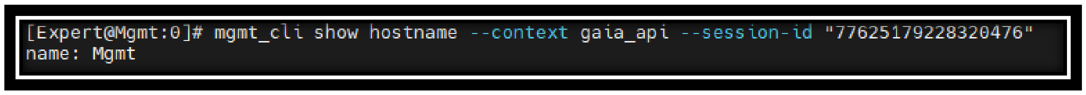
Parameter name	Short name	Environment variable
User name	-u	MGMT_CLI_USER
Password	-p	MGMT_CLI_PASSWORD
Check Point server address	-m	MGMT_CLI_MANAGEMENT

3. Login to the Gaia API using the command below and notice that we received a session ID (sid) back from the server.



```
[Expert@Mgmt:0]# mgmt_cli login -u api_admin -p vpn123 --context gaia_api
api-server-version: "1.6"
last-login-was-at:
  iso-8601: "2021-10-18T16:41+2000"
  posix: "1634589660209"
read-only: "false"
session-timeout: "600"
sid: "776251792283204 76"
url: "https://127.0.0.1:443/gaia_api"
```

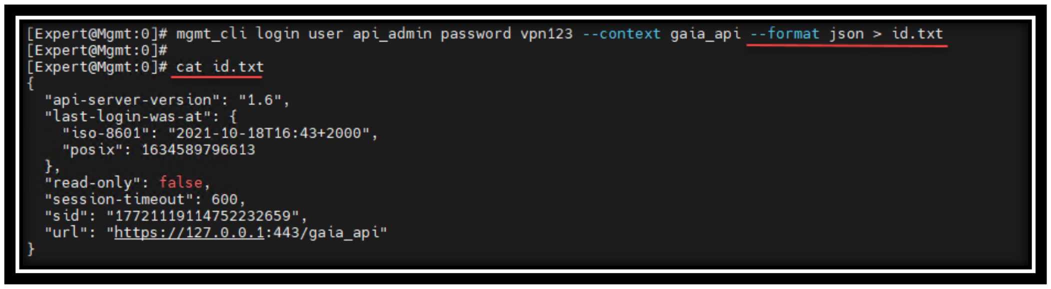
4. Run a new API call to get the hostname of the Gaia server using the session ID we received in the previous step.



```
[Expert@Mgmt:0]# mgmt_cli show hostname --context gaia_api --session-id "776251792283204 76"
name: Mgmt
```

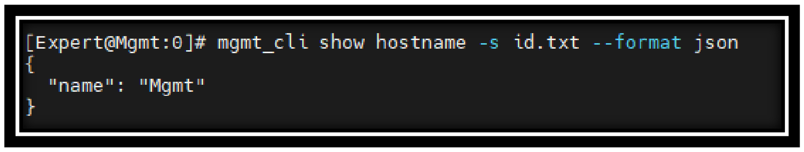
- Note that it is inconvenient to provide the session-id manually to each command. The proper way is to save the reply from the server to a text file where mgmt_cli can read the “SID”.

5. Use the command below to log in to the Gaia API server and save the results in a text file called id.txt. Note that we’re using the argument –format (you can also use –f to request the server to reply using JSON format)



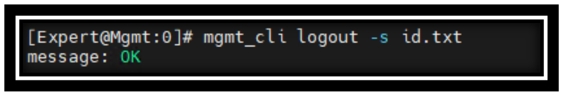
```
[Expert@Mgmt:0]# mgmt_cli login user api_admin password vpn123 --context gaia_api --format json > id.txt
[Expert@Mgmt:0]#
[Expert@Mgmt:0]# cat id.txt
{
  "api-server-version": "1.6",
  "last-login-was-at": {
    "iso-8601": "2021-10-18T16:43+2000",
    "posix": 1634589796613
  },
  "read-only": false,
  "session-timeout": 600,
  "sid": "17721119114752232659",
  "url": "https://127.0.0.1:443/gaia_api"
}
```

6. Use the mgmt_cli to show the hostname while reading the session ID from the file we saved in the previous step. Use the –format JSON to receive the reply in a JSON format.



```
[Expert@Mgmt:0]# mgmt_cli show hostname -s id.txt --format json
{
  "name": "Mgmt"
}
```

7. Finally, log out using the command below. This renders the session ID invalid and a login is required before issuing any new command.

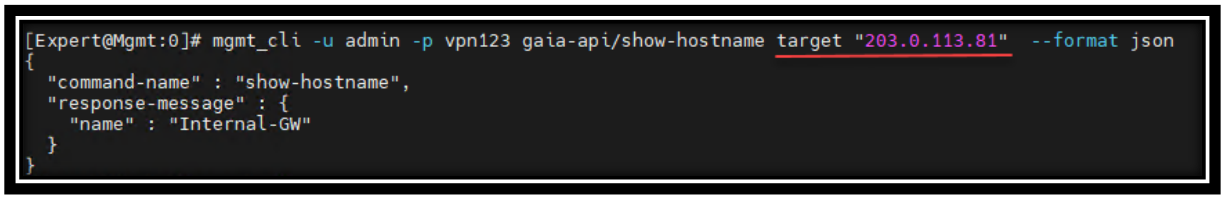


```
[Expert@Mgmt:0]# mgmt_cli logout -s id.txt
message: OK
```

Exercise 4: Gaia API via Management (Proxy)

In the previous exercise, we used mgmt_cli to access the Gaia API locally. The management Web API also uses mgmt_cli. In this exercise, we will learn how to use the management API as a proxy to run GAIA API commands against any target gateway.

1. From the Mobaxterm CLI, we have opened to the management station, run the following command and notice the error message.



```
[Expert@Mgmt:0]# mgmt_cli -u admin -p vpn123 gaia-api/show-hostname target "203.0.113.81"  --format json
{
  "command-name" : "show-hostname",
  "response-message" : {
    "name" : "Internal-GW"
  }
}
```

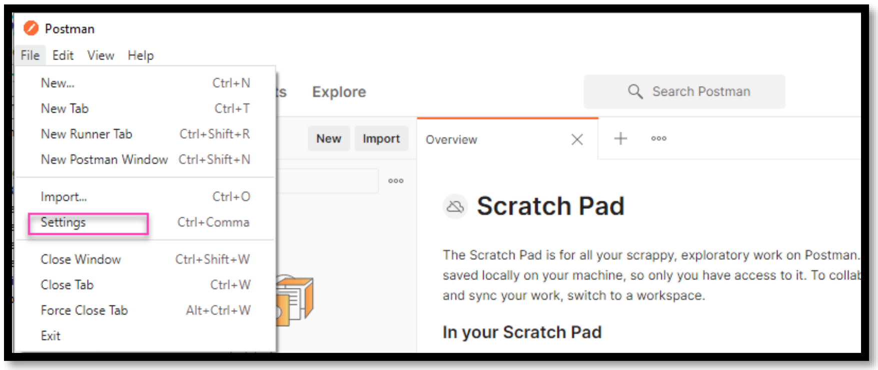
Notes:

- The credentials provided we used is the management administrator (not api_admin we created for the Gaia API).
- Instead of using the parameter "--context" we used the format Gaia-api/<command>.
- Using "--context," tells mgmt_cli that we are trying to use Gaia API.
- You can use "-m <address>" with mgmt_cli to log in to a remote host.
- We provided a required argument called "target" to specify where the management API should run the GAIA API command.

Exercise 5: Working with Postman

Up to this point, we used the mgmt_cli tool to run the Gaia API commands. In this exercise, we will use Postman to access the Gaia API.

1. From the Windows 10 Jumpbox, open Postman and open the settings windows.

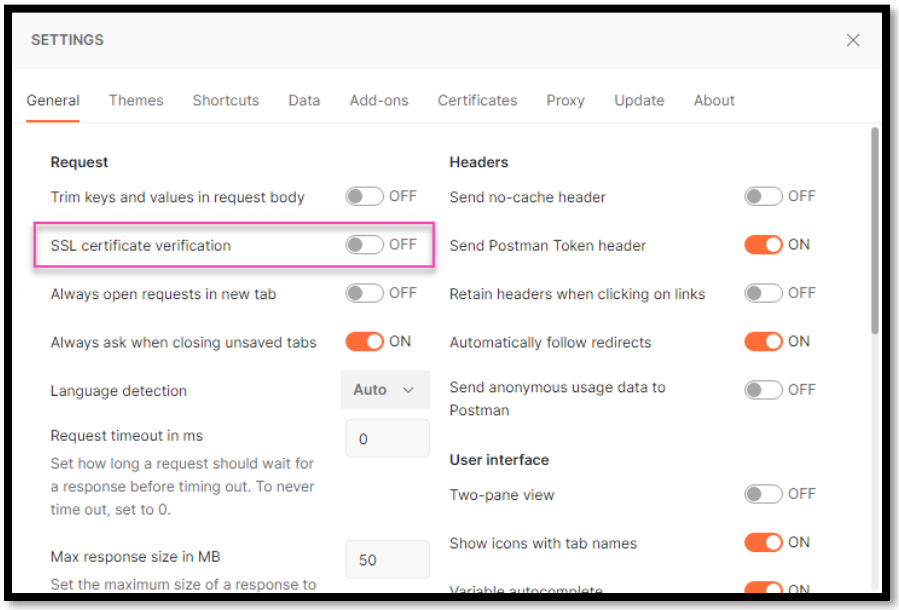


Scratch Pad

The Scratch Pad is for all your scrappy, exploratory work on Postman. saved locally on your machine, so only you have access to it. To collat and sync your work, switch to a workspace.

In your Scratch Pad

2. Disable the “SSL certificate verification”.



Request

Trim keys and values in request body	OFF
SSL certificate verification	OFF
Always open requests in new tab	OFF
Always ask when closing unsaved tabs	ON
Language detection	Auto
Request timeout in ms Set how long a request should wait for a response before timing out. To never time out, set to 0.	0
Max response size in MB Set the maximum size of a response to	50

Headers

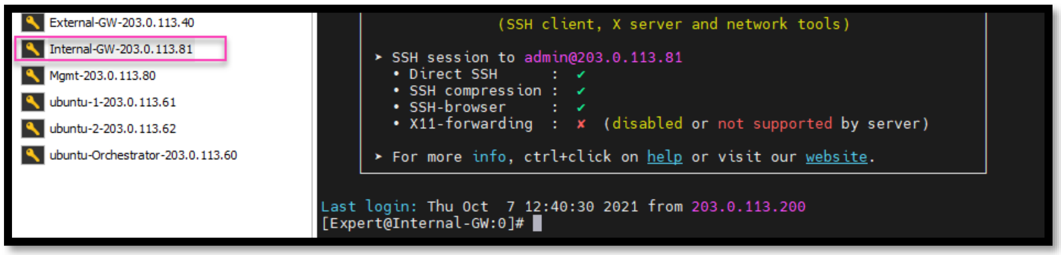
Send no-cache header	OFF
Send Postman Token header	ON
Retain headers when clicking on links	OFF
Automatically follow redirects	ON
Send anonymous usage data to Postman	OFF

User interface

Two-pane view	OFF
Show icons with tab names	ON
Variable autocomplete	ON

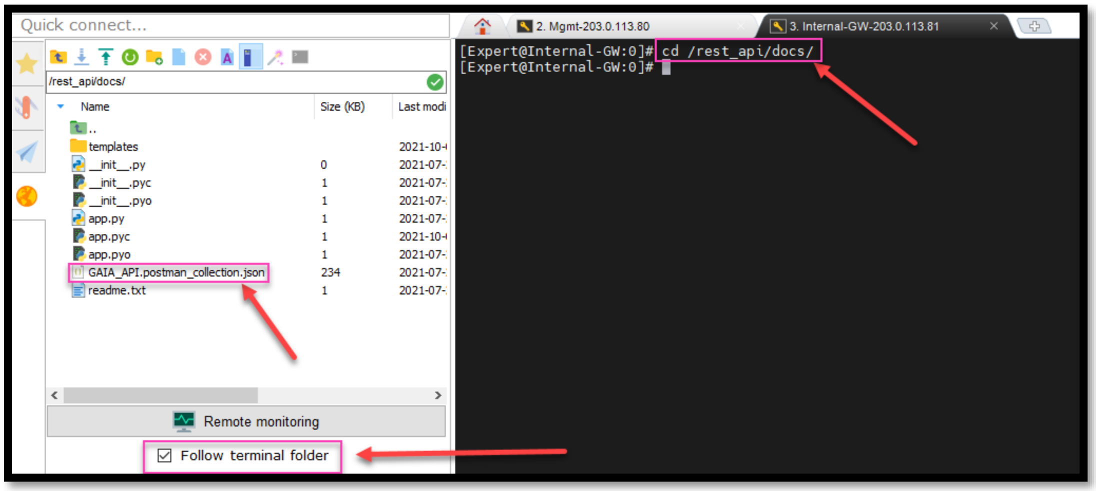
- Note: this step is necessary as we have a self-signed certificate on our Gaia servers.

3. Use MobaXterm to connect to the Internal-GW-203.0.113.81 over SSH.



```
                  (SSH client, X server and network tools)

  ➤ SSH session to admin@203.0.113.81
    • Direct SSH       :  ✔
    • SSH compression  :  ✔
    • SSH-browser      :  ✔
    • X11-forwarding   :  ✘  (disabled or not supported by server)

  ➤ For more info, ctrl+click on help or visit our website.

Last login: Thu Oct  7 12:40:30 2021 from 203.0.113.200
[Expert@Internal-GW:0]#
```

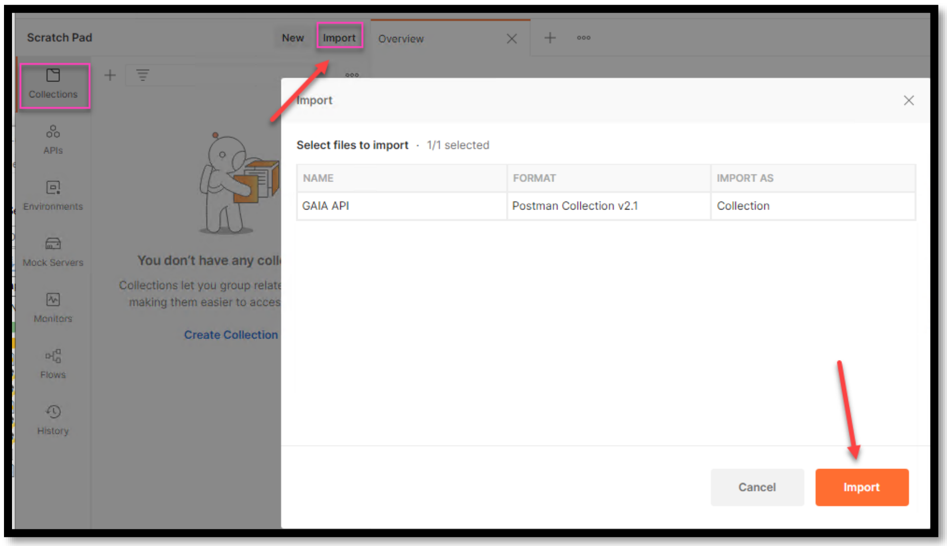
4. Make sure to enable “Follow terminal folder”, then run the command “cd /rest_api/docs/” to access the location where the Gaia API collection for postman is located by default.



Name	Size (KB)	Last modi
..		
templates		2021-10-
__init__.py	0	2021-07-
__init__.pyc	1	2021-07-
__init__.pyo	1	2021-07-
app.py	1	2021-07-
app.pyc	1	2021-10-
app.pyo	1	2021-07-
GAIA_API.postman_collection.json	234	2021-07-
readme.txt	1	2021-07-

```
[Expert@Internal-GW:0]# cd /rest_api/docs/
[Expert@Internal-GW:0]#
```

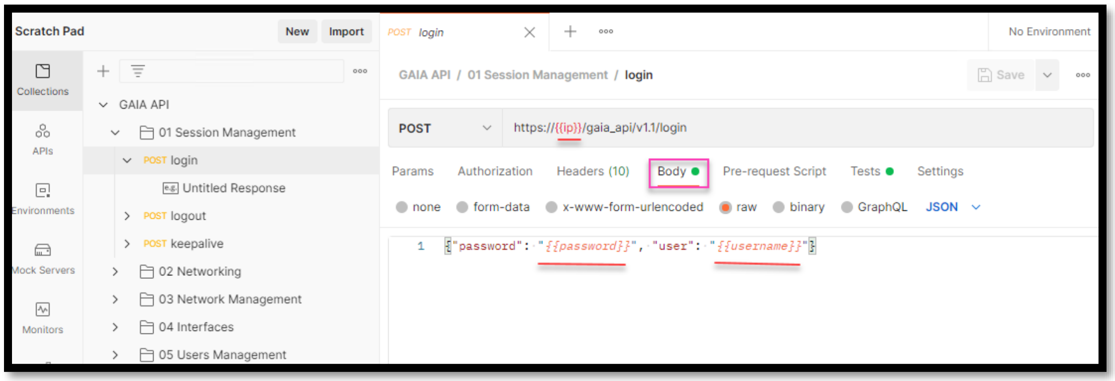
5. Drag and drop the file to your desktop.
6. Import the collection from the file as you can see in the screenshot below.



Select files to import · 1/1 selected

NAME	FORMAT	IMPORT AS
GAIA API	Postman Collection v2.1	Collection

7. Expand the Gaia API list and under “01 Session Management” select “POST Login” and switch to the “Body tab”.

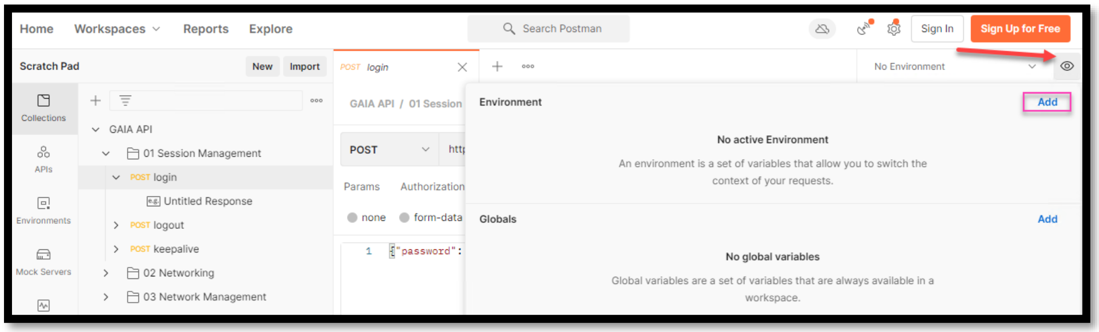


```
POST  https://{{ip}}/gaia_api/v1.1/login
```

```
{"password": "{{password}}", "user": "{{username}}"}
```

- Note that the IP, user, and password were created as variables.
- You can replace that manually but the proper way is to create an environment that contains the values of those variables.
- Remember that each call has the same {{ip}} variable and manually replacing it with your address is not logical.

8. Click in the environment icon seen in the screenshot below to create a new environment and click “Add”



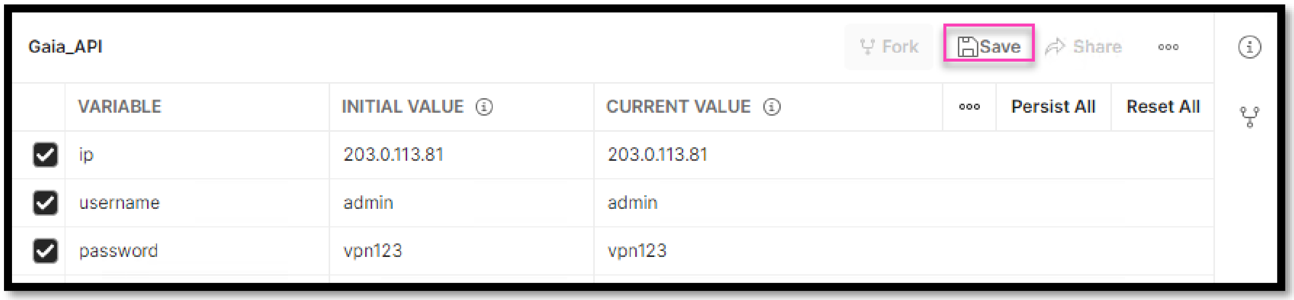
No active Environment

An environment is a set of variables that allow you to switch the context of your requests.

No global variables

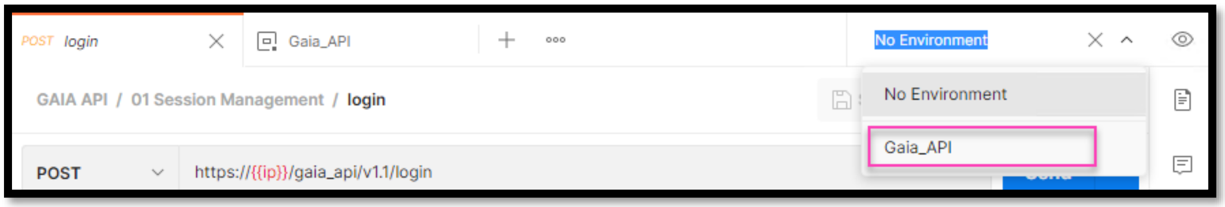
Global variables are a set of variables that are always available in a workspace.

9. Fill in the details to provide values for those variables and click Save.



	VARIABLE	INITIAL VALUE	CURRENT VALUE
✓	ip	203.0.113.81	203.0.113.81
✓	username	admin	admin
✓	password	vpn123	vpn123

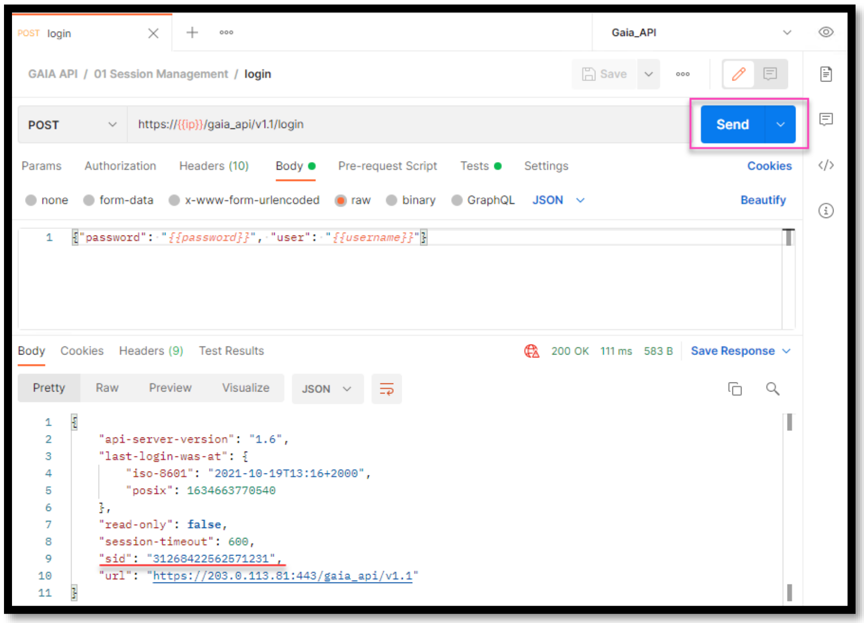
10. Select the environment we just created



```
https://{{ip}}/gaia_api/v1.1/login
```

- Tip: Hover the mouse over the {{ip}} variable and make sure you can see the real address.

11. Run the login command by clicking "Send" and review the results.



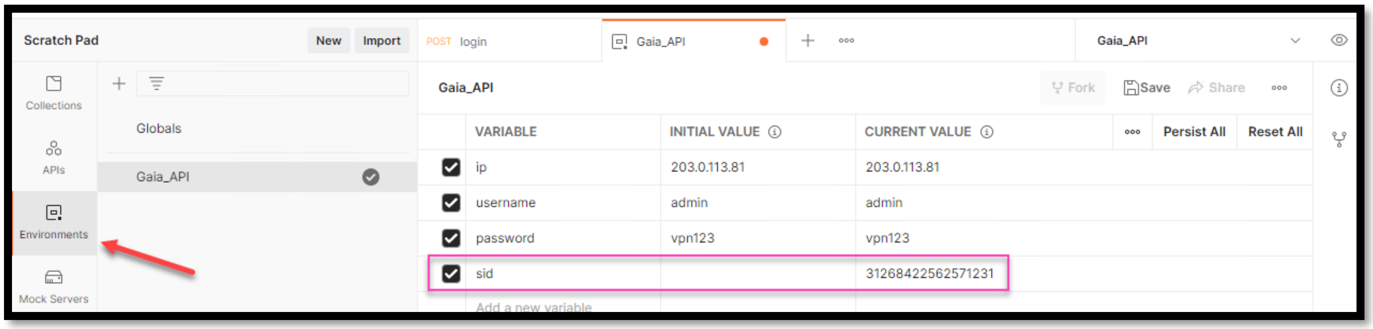
```
https://{{ip}}/gaia_api/v1.1/login
```

```
{"password": "{{password}}", "user": "{{username}}"}
```

200 OK 111 ms 583 B

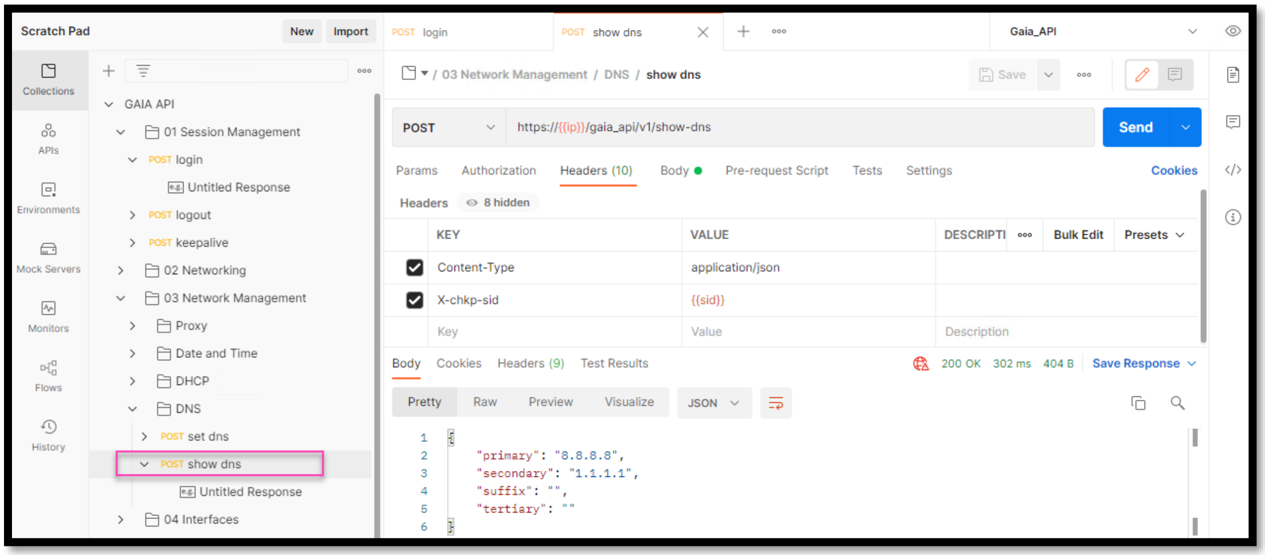
```
{
    "api-server-version": "1.6",
    "last-login-was-at": {
        "iso-8601": "2021-10-19T13:16+2000",
        "posix": 1634663770540
    },
    "read-only": false,
    "session-timeout": 600,
    "sid": "31268422562571231",
    "url": "https://203.0.113.81:443/gaia_api/v1.1"
}
```

12. Under "Environments", select the Gaia_API environment we created in the previous steps and notice that we now have the session ID "sid" saved.



	VARIABLE	INITIAL VALUE	CURRENT VALUE
✓	ip	203.0.113.81	203.0.113.81
✓	username	admin	admin
✓	password	vpn123	vpn123
✓	sid		31268422562571231

13. Now that we are connected and have a session, we should be able to use any other call. Run a call to show the existing DNS server.

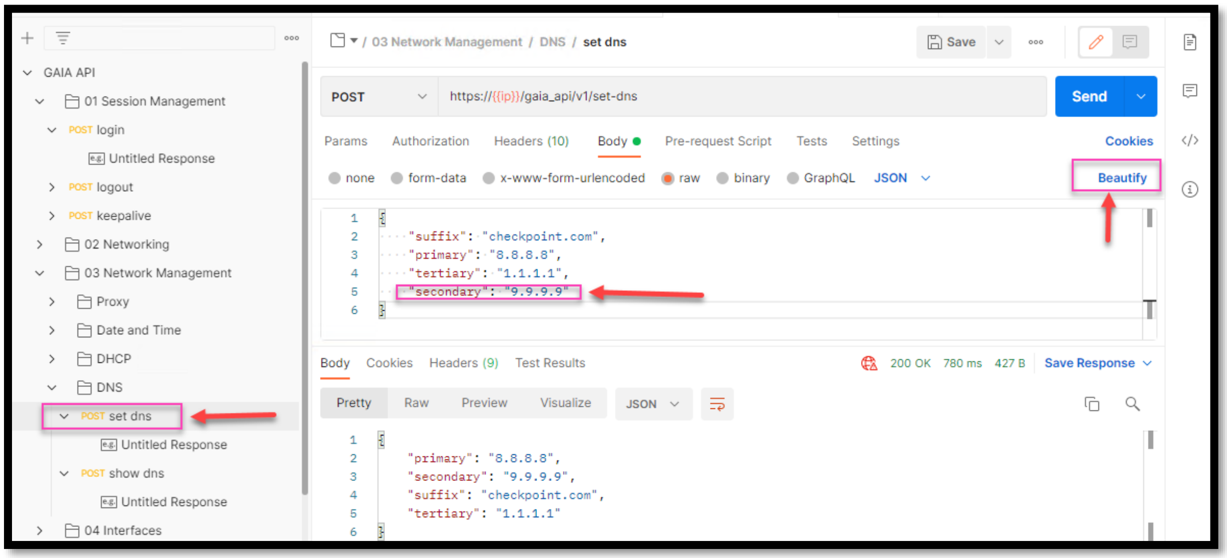


POST https://{{ip}}/gaia_api/v1/show-dns

KEY	VALUE	DESCRIPTION
✓ Content-Type	application/json	
✓ X-chkp-sid	{{sid}}	
Key	Value	Description

```
{
    "primary": "8.8.8.8",
    "secondary": "1.1.1.1",
    "suffix": "",
    "tertiary": ""
}
```

14. Use the saved call "set DNS" to add a third DNS server. Use the "Beautify" feature shown below to show the call in proper JSON format.



POST https://{{ip}}/gaia_api/v1/set-dns

```
{
    "suffix": "checkpoint.com",
    "primary": "8.8.8.8",
    "tertiary": "1.1.1.1",
    "secondary": "9.9.9.9"
}
```

```
{
    "primary": "8.8.8.8",
    "secondary": "9.9.9.9",
    "suffix": "checkpoint.com",
    "tertiary": "1.1.1.1"
}
```

15. Run a few calls to get familiar with the required fields, header, and body for different calls.
16. Once you feel familiar with the commands, log out of the Gaia API.

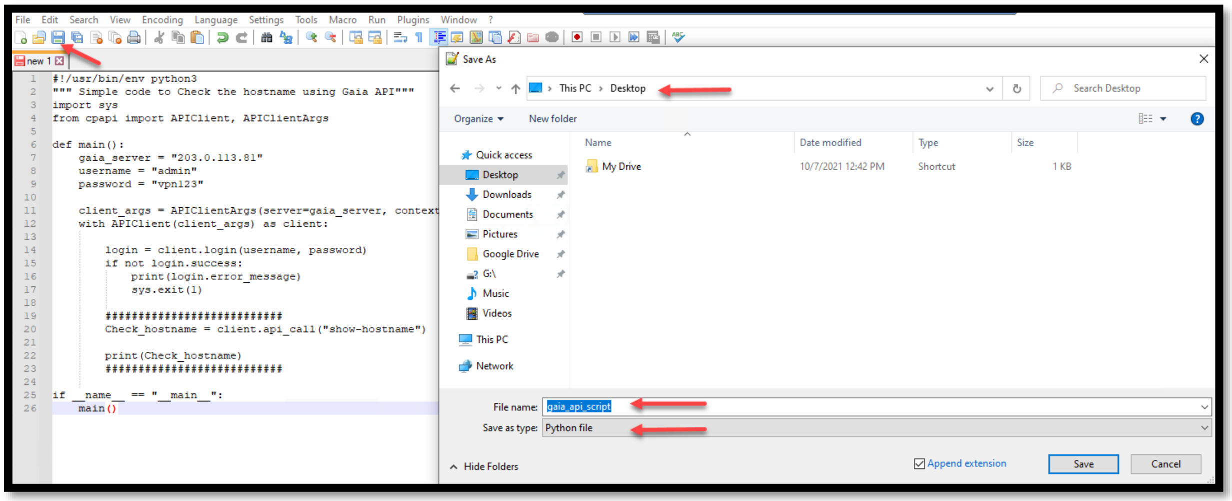
Exercise 6: Using the Check Point Python SDK

In this exercise, we will use the Check Point python SDK to write a simple script to run the Gaia API command and manage our Gaia servers. The SDK and its documentation are available on GitHub (https://github.com/CheckPointSW/cp_mgmt_api_python_sdk).

1. Open Notepad++ and paste the code template below to the file or type it if you prefer or download from (https://github.com/alshawwaf/Gaia_API-Workshop_2021.git).

```python
#!/usr/bin/env python3

""" Simple code to Check the hostname using Gaia API"""

import sys
from cpapi import APIClient, APIClientArgs

def main():
    gaia_server = "203.0.113.81"
    username = "admin"
    password = "vpn123"

    client_args = APIClientArgs(server=gaia_server, context='gaia_api')
    with APIClient(client_args) as client:

        login = client.login(username, password)
        if not login.success:
            print(login.error_message)
            sys.exit(1)

        ###########################

        Check_hostname = client.api_call("show-hostname")

        print(Check_hostname)

        ###########################

if __name__ == "__main__":
    main()
```

2. Save this file to your desktop and make sure the file type is set to “Python” as you can see below.

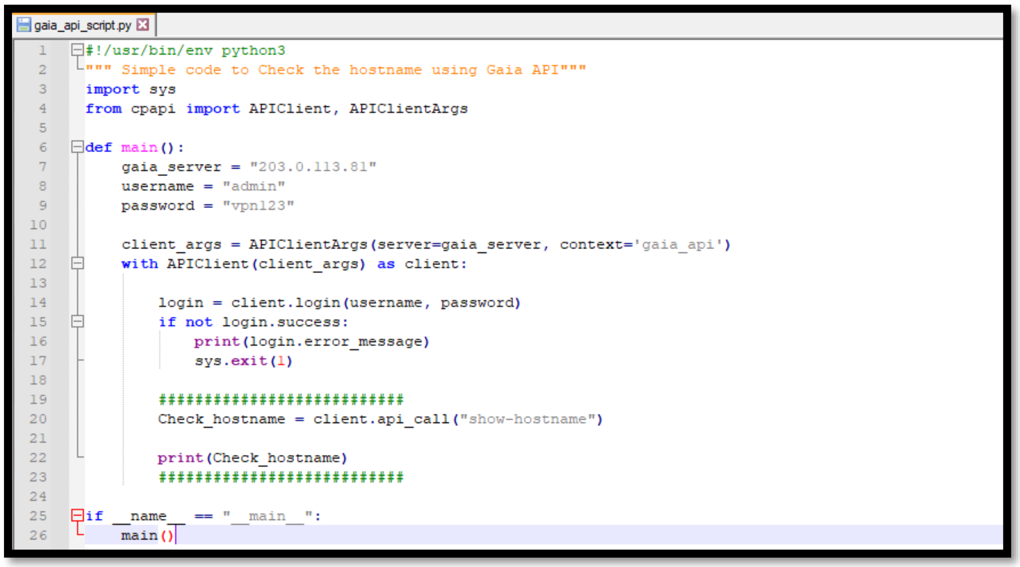


```
#!/usr/bin/env python3
""" Simple code to Check the hostname using Gaia API"""
import sys
from cpapi import APIClient, APIClientArgs

def main():
    gaia_server = "203.0.113.81"
    username = "admin"
    password = "vpn123"

    client_args = APIClientArgs(server=gaia_server, context
    with APIClient(client_args) as client:

        login = client.login(username, password)
        if not login.success:
            print(login.error_message)
            sys.exit(1)

        ###########################
        Check_hostname = client.api_call("show-hostname")

        print(Check_hostname)
        ###########################

if __name__ == "__main__":
    main()
```

File name: gaia_api_script

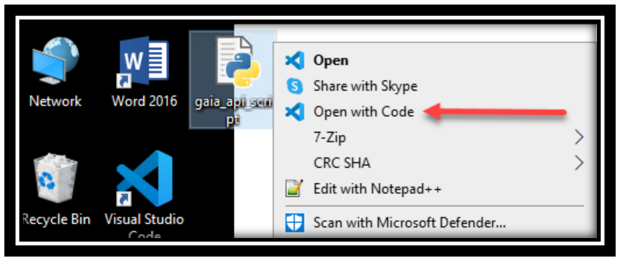
Save as type: Python file

- Notice that Notepad++ is now showing the python script in a better format.

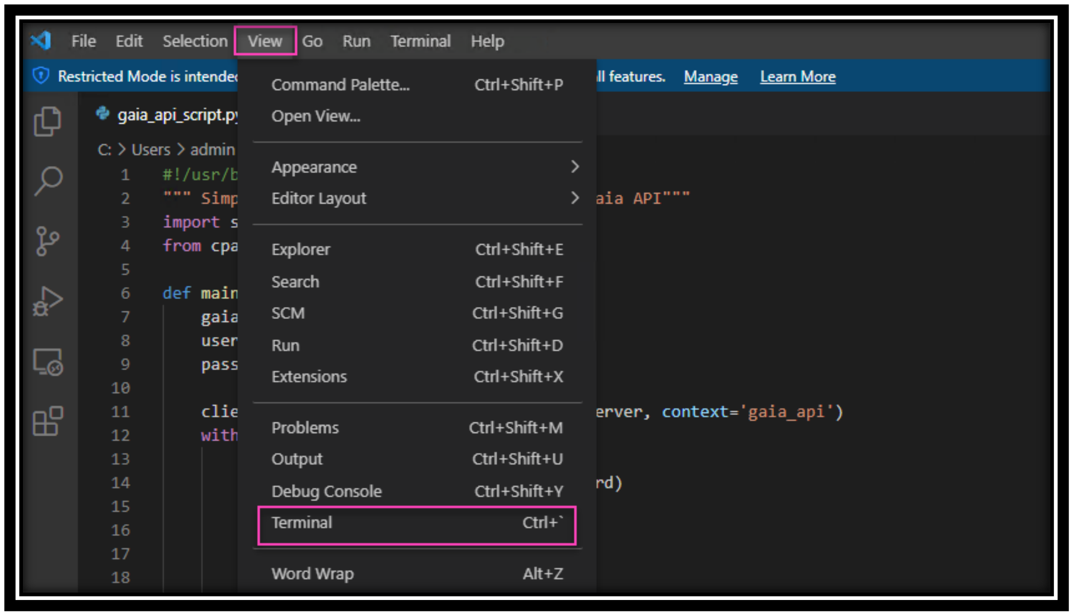


```
#!/usr/bin/env python3
""" Simple code to Check the hostname using Gaia API"""
import sys
from cpapi import APIClient, APIClientArgs

def main():
    gaia_server = "203.0.113.81"
    username = "admin"
    password = "vpn123"

    client_args = APIClientArgs(server=gaia_server, context='gaia_api')
    with APIClient(client_args) as client:

        login = client.login(username, password)
        if not login.success:
            print(login.error_message)
            sys.exit(1)

        ###########################
        Check_hostname = client.api_call("show-hostname")

        print(Check_hostname)
        ###########################

if __name__ == "__main__":
    main()
```

3. It is preferred to use a more advanced IDE such as VScode. Close your notepad++. Right-Click on the file from the desktop and select "Open with Code" to open the file using Visual Studio Code.



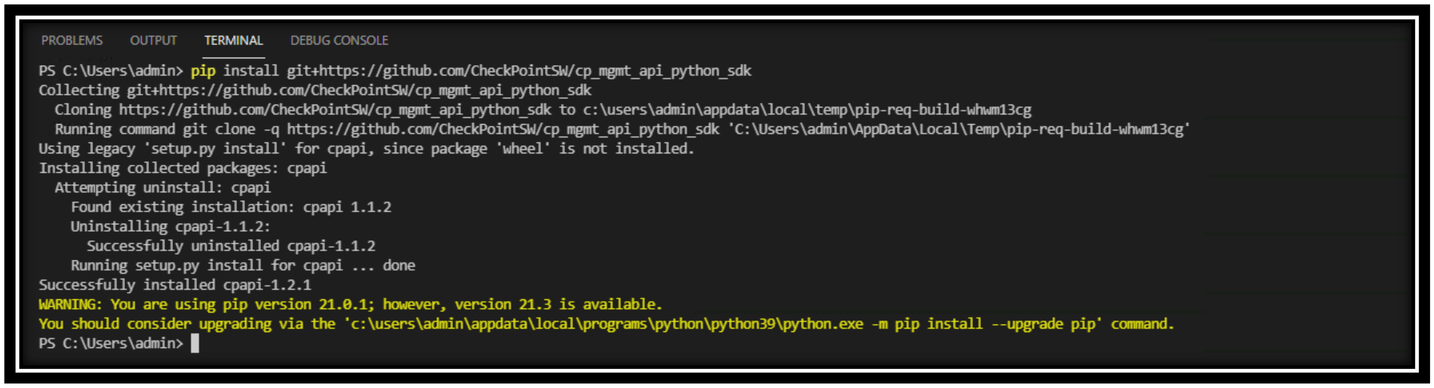
4. VSCode provides the ability to run scripts using integrated terminals. From the View tab, select “terminal to open the default terminal or use (Ctrl+`).



```
#!/usr/b
""" Simp                                  aia API"""
import s
from cpa

def main
    gaia
    user
    pass

    clie                                  erver, context='gaia_api')
    with

                                          rd)
```

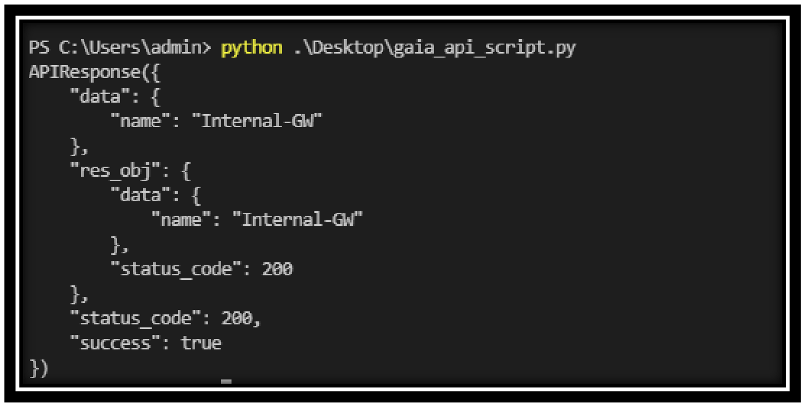
5. Use the terminal to install the Check Point Python SDK using the command below:

pip install git+https://github.com/CheckPointSW/cp_mgmt_api_python_sdk

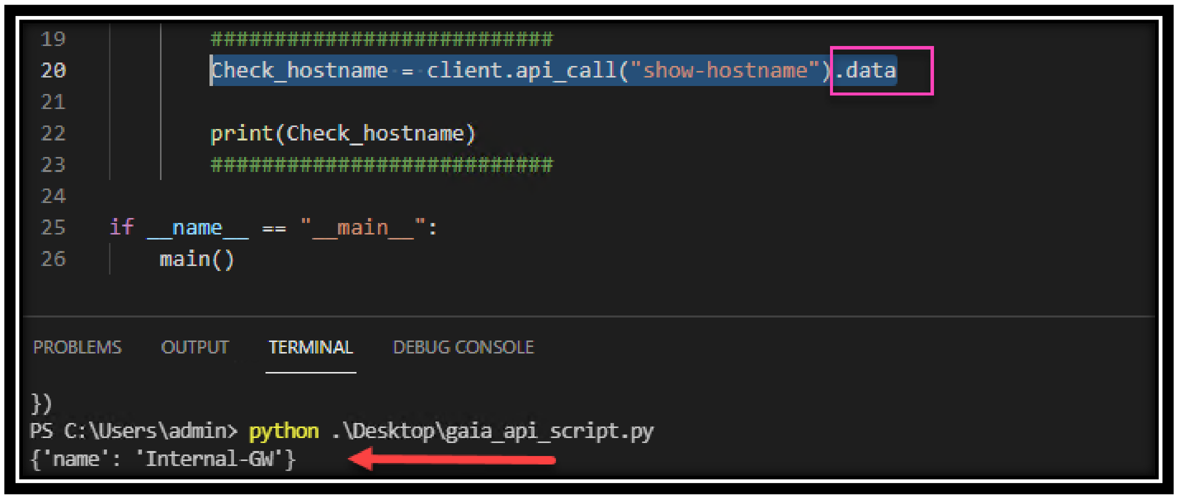


```
PS C:\Users\admin> pip install git+https://github.com/CheckPointSW/cp_mgmt_api_python_sdk
Collecting git+https://github.com/CheckPointSW/cp_mgmt_api_python_sdk
  Cloning https://github.com/CheckPointSW/cp_mgmt_api_python_sdk to c:\users\admin\appdata\local\temp\pip-req-build-whwm13cg
  Running command git clone -q https://github.com/CheckPointSW/cp_mgmt_api_python_sdk 'C:\Users\admin\AppData\Local\Temp\pip-req-build-whwm13cg'
Using legacy 'setup.py install' for cpapi, since package 'wheel' is not installed.
Installing collected packages: cpapi
  Attempting uninstall: cpapi
    Found existing installation: cpapi 1.1.2
    Uninstalling cpapi-1.1.2:
      Successfully uninstalled cpapi-1.1.2
    Running setup.py install for cpapi ... done
Successfully installed cpapi-1.2.1
WARNING: You are using pip version 21.0.1; however, version 21.3 is available.
You should consider upgrading via the 'c:\users\admin\appdata\local\programs\python\python39\python.exe -m pip install --upgrade pip' command.
PS C:\Users\admin> 
```

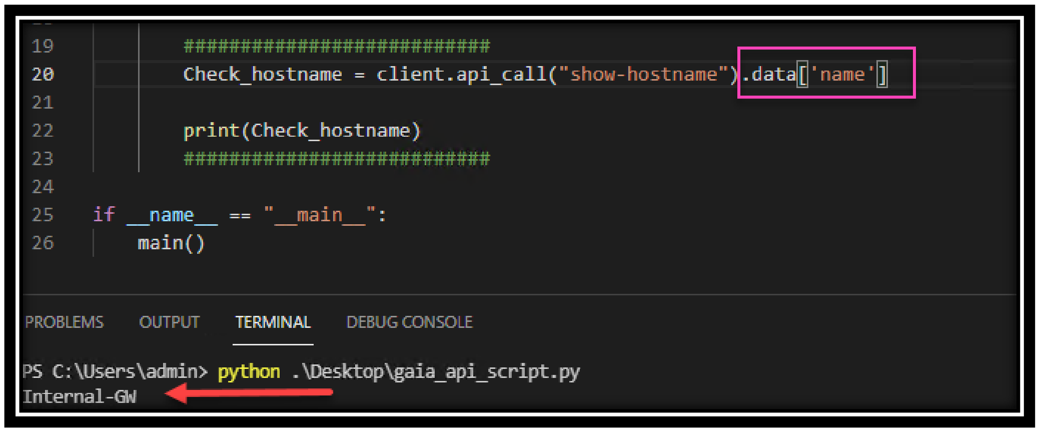
6. Now we should be able to run our script. Use the command “python .\Desktop\gaia_api_script.py” to run the script and accept the fingerprints if you see a prompt.


```
PS C:\Users\admin> python .\Desktop\gaia_api_script.py
APIResponse({
    "data": {
        "name": "Internal-GW"
    },
    "res_obj": {
        "data": {
            "name": "Internal-GW"
        },
        "status_code": 200
    },
    "status_code": 200,
    "success": true
})
```

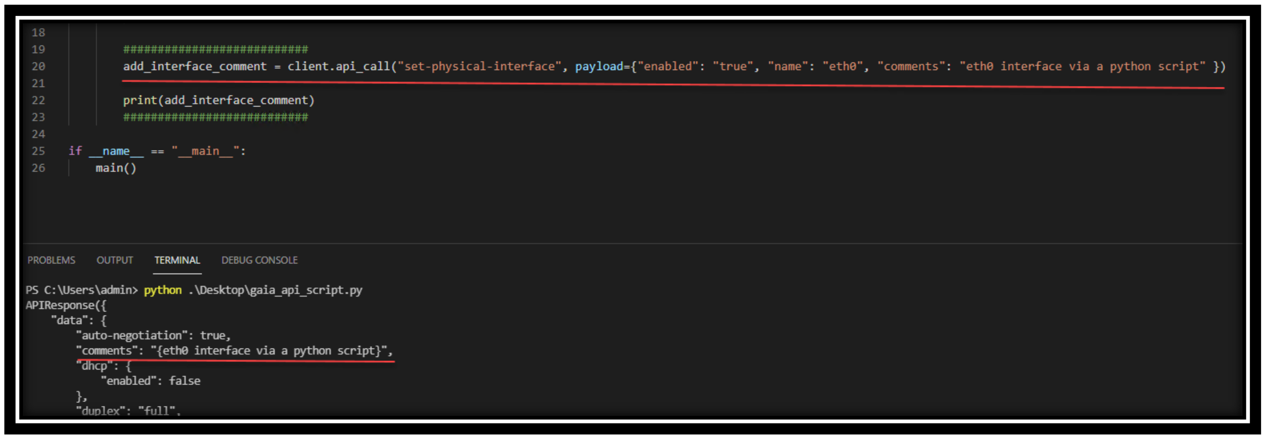
7. Notice that we can filter the reply to show only the requested field without the status codes. Change the check_hostname call to add .data at the end “Check_hostname = client.api_call("show-hostname").data” This will only show fields inside the data object.


```
        ###########################
        Check_hostname = client.api_call("show-hostname").data

        print(Check_hostname)
        ###########################

if __name__ == "__main__":
    main()
```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
```
})
PS C:\Users\admin> python .\Desktop\gaia_api_script.py
{'name': 'Internal-GW'}
```

8. We can also filter using the key of the JSON reply to show only the value.


```
        ###########################
        Check_hostname = client.api_call("show-hostname").data['name']

        print(Check_hostname)
        ###########################

if __name__ == "__main__":
    main()
```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
```
PS C:\Users\admin> python .\Desktop\gaia_api_script.py
Internal-GW
```

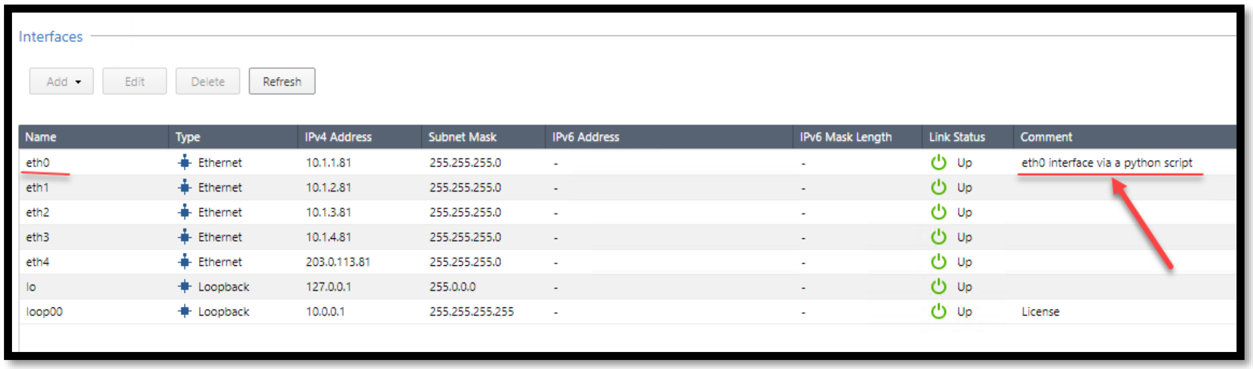
9. Change your script to edit the interface “eth0” and add a comment to this interface. Consult with the Gaia API documentation to find the required fields. Notice that we are providing the required fields as payload.



```
        ###########################
        add_interface_comment = client.api_call("set-physical-interface", payload={"enabled": "true", "name": "eth0", "comments": "eth0 interface via a python script" })

        print(add_interface_comment)
        ###########################

if __name__ == "__main__":
    main()
```

```
PS C:\Users\admin> python .\Desktop\gaia_api_script.py
APIResponse({
    "data": {
        "auto-negotiation": true,
        "comments": "{eth0 interface via a python script}",
        "dhcp": {
            "enabled": false
        },
        "duplex": "full",
```

10. Review the interface settings via the Gaia Portal and make sure you can see the comment.



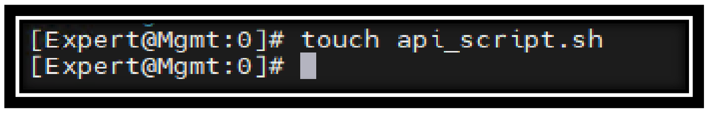
Name	Type	IPv4 Address	Subnet Mask	IPv6 Address	IPv6 Mask Length	Link Status	Comment
eth0	Ethernet	10.1.1.81	255.255.255.0	-	-	Up	eth0 interface via a python script
eth1	Ethernet	10.1.2.81	255.255.255.0	-	-	Up	
eth2	Ethernet	10.1.3.81	255.255.255.0	-	-	Up	
eth3	Ethernet	10.1.4.81	255.255.255.0	-	-	Up	
eth4	Ethernet	203.0.113.81	255.255.255.0	-	-	Up	
lo	Loopback	127.0.0.1	255.0.0.0	-	-	Up	
loop00	Loopback	10.0.0.1	255.255.255.255	-	-	Up	License

11. Exit the VScode.

Exercise 7: Writing bash script with mgmt_cli

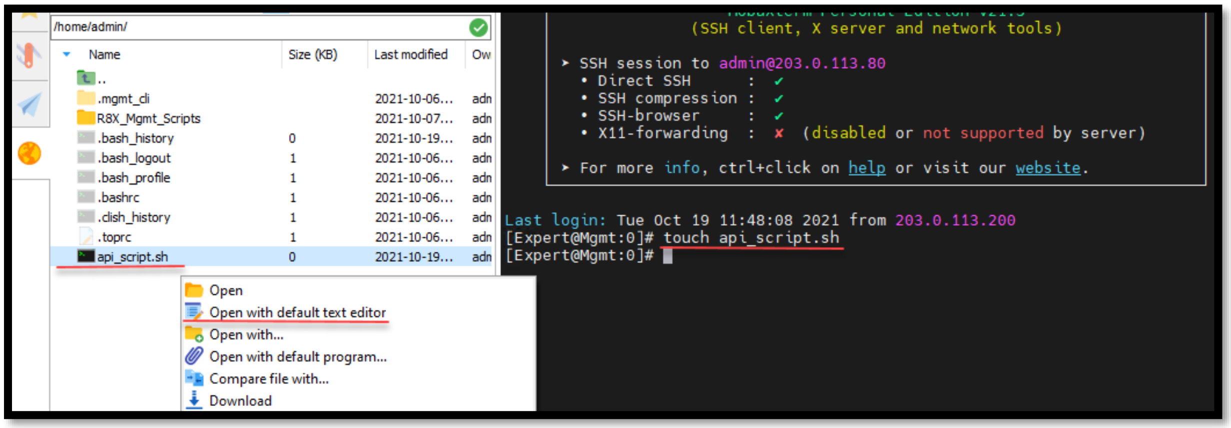
In this exercise, we will write simple bash scripts to utilize the mgmt_cli and perform tasks against the Gaia servers.

1. Connect to the CLI of the management server using Mobaxterm and create an empty file.



```
[Expert@Mgmt:0]# touch api_script.sh
[Expert@Mgmt:0]# 
```

2. Use any text editor to edit the script file such as "vi". Alternatively, you can open it using the default text editor of MobaXterm

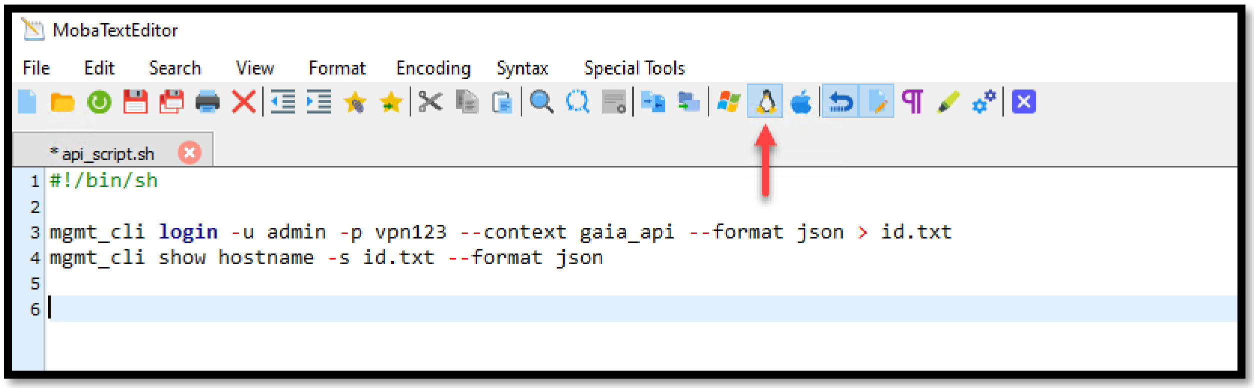

/home/admin/
Name
Size (KB)
Last modified
Ow
..
.mgmt_cli 2021-10-06... adn
R8X_Mgmt_Scripts 2021-10-07... adn
.bash_history 0 2021-10-19... adn
.bash_logout 1 2021-10-06... adn
.bash_profile 1 2021-10-06... adn
.bashrc 1 2021-10-06... adn
.clish_history 1 2021-10-06... adn
.toprc 1 2021-10-06... adn
api_script.sh 0 2021-10-19... adn
Open
Open with default text editor
Open with...
Open with default program...
Compare file with...
Download

```
(SSH client, X server and network tools)

 ➤ SSH session to admin@203.0.113.80
   • Direct SSH      :  ✓
   • SSH compression :  ✓
   • SSH-browser     :  ✓
   • X11-forwarding  :  ✘  (disabled or not supported by server)

 ➤ For more info, ctrl+click on help or visit our website.

Last login: Tue Oct 19 11:48:08 2021 from 203.0.113.200
[Expert@Mgmt:0]# touch api_script.sh
[Expert@Mgmt:0]#
```

3. Copy the script lines below or type them if you prefer in the file we just created.

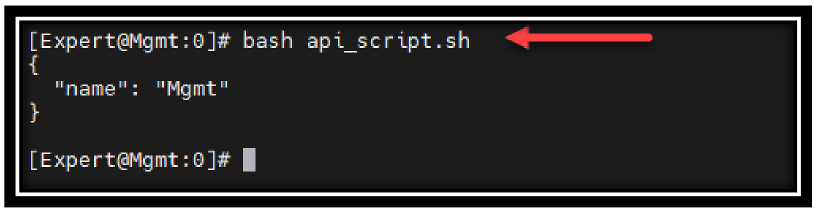
```
#!/bin/sh

mgmt_cli login -u admin -p vpn123 --context gaia_api --format json > id.txt
mgmt_cli show hostname -s id.txt --format json
```

- Note that if you use the built-in editor, you must select "UNIX format" as shown below.



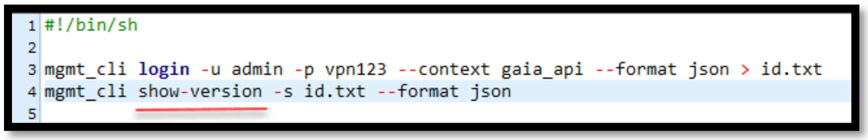
```
#!/bin/sh

mgmt_cli login -u admin -p vpn123 --context gaia_api --format json > id.txt
mgmt_cli show hostname -s id.txt --format json


```

4. Run your script from the command line using the command "bash gaia_api_script.sh"



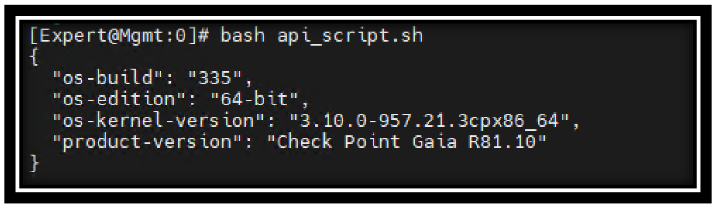
```
[Expert@Mgmt:0]# bash api_script.sh
{
  "name": "Mgmt"
}

[Expert@Mgmt:0]#
```

5. Change the script to perform a call to get the version details of the Gaia server.



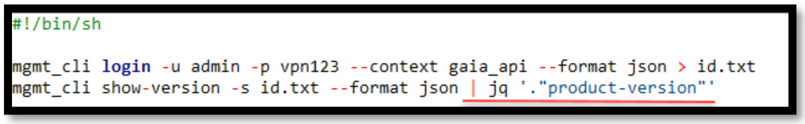
```
#!/bin/sh

mgmt_cli login -u admin -p vpn123 --context gaia_api --format json > id.txt
mgmt_cli show-version -s id.txt --format json

```

6. Run the script and examine the output details.



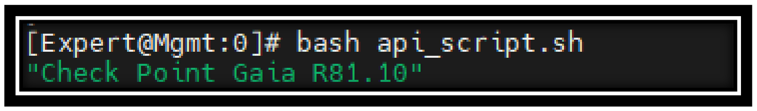
```
[Expert@Mgmt:0]# bash api_script.sh
{
  "os-build": "335",
  "os-edition": "64-bit",
  "os-kernel-version": "3.10.0-957.21.3cpx86_64",
  "product-version": "Check Point Gaia R81.10"
}
```

7. Check Point Gaia has the JQ utility installed on all supported versions. This tool can help us filter the JSON response and have the exact field. Change the script and use “jq” to filter the response to only show the “product-version”



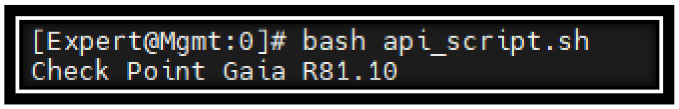
```
#!/bin/sh

mgmt_cli login -u admin -p vpn123 --context gaia_api --format json > id.txt
mgmt_cli show-version -s id.txt --format json | jq '."product-version"'
```

8. Run the script and notice that only the product version was returned.



```
[Expert@Mgmt:0]# bash api_script.sh
"Check Point Gaia R81.10"
```

- Note: use the flag “-r” with JQ to get rid of the quotes.



```
[Expert@Mgmt:0]# bash api_script.sh
Check Point Gaia R81.10
```

Thank You