QUANTUM

15 December 2024

SKYLINE

Administration Guide

CHECK POINT™

Check Point Copyright Notice

© 2022 - 2024 Check Point Software Technologies Ltd.

All rights reserved. This product and related documentation are protected by copyright and distributed under licensing restricting their use, copying, distribution, and decompilation. No part of this product or related documentation may be reproduced in any form or by any means without prior written authorization of Check Point. While every precaution has been taken in the preparation of this book, Check Point assumes no responsibility for errors or omissions. This publication and features described herein are subject to change without notice.

RESTRICTED RIGHTS LEGEND:

Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

TRADEMARKS:

Refer to the Copyright page for a list of our trademarks.

Refer to the Third Party copyright notices for a list of relevant copyrights and third-party licenses.

Important Information

Latest Software

We recommend that you install the most recent software release to stay up-to-date with the latest functional improvements, stability fixes, security enhancements and protection against new and evolving attacks.

Certifications

For third party independent certification of Check Point products, see the Check Point Certifications page.

Check Point Skyline Administration Guide

For more about this product, see the home page.

Latest Version of this Document in English

Open the latest version of this document in a Web browser.
Download the latest version of this document in PDF format.

Feedback

Check Point is engaged in a continuous effort to improve its documentation.
Please help us by sending your comments.

Revision History

Date	Description
12 December 2024	Added: ■ *"Custom Metrics" on page 265* ■ *"API on the Management Server" on page 238* ■ *"Blades > Status and Update" on page 188* ■ *"Maestro Orchestrator" on page 242* ■ *"Network > Heavy Connections" on page 172* ■ *"Network > Network Probes (VPN)" on page 178* ■ *"OtlpAgent > Scripts" on page 230* ■ *"OtlpAgent > Version" on page 237* ■ *"System > CPU > Top" on page 68* ■ *"System > Process > Top" on page 144* ■ *"System > Process" on page 148* ■ *"VoIP" on page 169* ■ *"VPN > Probes" on page 182* Updated: ■ Improved formatting ■ *"Skyline Configuration on Check Point Servers that run Gaia OS - Prometheus with Grafana" on page 17* ■ *"Skyline Configuration on Check Point Servers that run Gaia OS - Other Monitoring Tools" on page 38* ■ *"Hardware > BIOS" on page 228* ■ *"Hardware > PSU" on page 224* ■ *"System > CoreXL" on page 127* ■ *"System > Gaia" on page 123* ■ *"System > SecureXL > SYN Defender" on page 134*
03 May 2024	Added: ■ *"Skyline Configuration on Check Point Servers that run Gaia OS - Other Monitoring Tools" on page 38* Updated: ■ *"Introduction" on page 14* ■ *"Skyline Configuration on Check Point Servers that run Gaia OS - Prometheus with Grafana" on page 17* ■ *"Skyline Configuration on Quantum Spark Appliances" on page 51*

Date	Description
01 May 2024	Changed this book from "Skyline Metrics Repository" to "Skyline Administration Guide". Added: ■ *"Skyline Configuration on Check Point Servers that run Gaia OS - Prometheus with Grafana" on page 17* ■ *"Skyline Configuration on Quantum Spark Appliances" on page 51*
17 March 2024	Updated: ■ *"System > SecureXL > SYN Defender" on page 134*
20 November 2023	Added a new topic: ■ *"System > SecureXL > SYN Defender" on page 134*
31 August 2023	Updated: ■ *"System > Network > Interfaces" on page 87*
16 August 2023	Updated: ■ *"Blades > VPN" on page 194* - removed the metric "`cluster_xl.mac_magic`" (because it is not supported anymore)
29 May 2023	Updated: ■ *"ClusterXL" on page 160*
15 March 2023	Updated: ■ *"Blades > IDA" on page 210* ■ *"Hardware > BIOS" on page 228* ■ *"Hardware > Fans" on page 226* ■ *"Hardware > Model" on page 219* ■ *"Hardware > PSU" on page 224* ■ *"Hardware > Temperature" on page 220* ■ *"Hardware > Voltage" on page 222* ■ *"System > Flofiler" on page 139*
24 October 2022	Updated document for General Availability (GA) Release

Date	Description
07 April 2022	Updated: ■ CPView screenshots ■ *"System > Memory" on page 71* ■ *"System > Network" on page 84* ■ *"System > Filesystem" on page 119*
05 April 2022	First release of this document

Table of Contents

Introduction ... *14*

Skyline Architecture ... *14*

Skyline Requirements ... *15*

Skyline Downloads ... *15*

Skyline Known Limitations ... *15*

Skyline Configuration ... *16*

Skyline Troubleshooting and FAQ ... *16*

Skyline Metrics ... *16*

Skyline Configuration on Check Point Servers that run Gaia OS - Prometheus with Grafana ... *17*

Video Tutorial ... *17*

Step 1 - Install the Prometheus Server ... *18*

Step 2 - Install the Grafana Server ... *21*

Step 3 - Install the OpenTelemetry Agent and OpenTelemetry Collector on the Check Point Server ... *25*

Step 4 - Configure the OpenTelemetry Collector on the Check Point Server to work with the Prometheus Server ... *26*

Step 5 - Configure the filter for the OpenTelemetry Collector exported metrics ... *35*

Step 6 - Configure Access Control Policy ... *37*

Skyline Configuration on Check Point Servers that run Gaia OS - Other Monitoring Tools ... *38*

Step 1 - Install the Third-Party Monitoring Tool ... *39*

Step 2 - Install the OpenTelemetry Agent and OpenTelemetry Collector on the Check Point Server ... *39*

Step 3 - Configure the OpenTelemetry Collector on the Check Point Server to work with the Third-Party Monitoring Tool ... *40*

Step 4 - Configure the filter for the OpenTelemetry Collector exported metrics ... *48*

Step 5 - Configure Access Control Policy ... *50*

Skyline Configuration on Quantum Spark Appliances ... *51*

Video Tutorial ... *51*

Step 1 - Install the Prometheus Server ... 51
Step 2 - Install the Grafana Server ... 55
Step 3 - Install the OpenTelemetry Agent and OpenTelemetry Collector on the Quantum Spark Appliance ... 59
Step 4 - Configure OpenTelemetry Collector on the Quantum Spark Appliance to work with the Prometheus Server ... 60
Step 5 - Configure Access Policy ... 63
Skyline Metrics Repository ... 64
System > CPU ... 65
CPView Gauges ... 65
CLI ... 65
Metric Information ... 66
System > CPU > Top ... 68
CPView Gauges ... 68
CLI ... 68
Metric Information ... 69
System > Memory ... 71
CPView Gauges ... 71
CLI ... 71
Metric Information ... 72
System > Memory Paging ... 76
CPView Gauges ... 76
CLI ... 76
Metric Information ... 76
System > Traffic ... 78
CPView Gauges ... 78
CLI ... 78
Metric Information ... 79
System > Network ... 84
CPView Gauges ... 84
CLI ... 84

Metric Information ... 85
System > Network > Interfaces ... 87
CPView Gauges ... 87
CLI ... 87
Metric Information ... 88
System > Network > Packets ... 101
CPView Gauges ... 101
CLI ... 102
Metric Information ... 103
System > Network > NAT ... 111
CPView Gauges ... 111
CLI ... 111
Metric Information ... 112
System > Filesystem ... 119
CPView Gauges ... 119
CLI ... 119
Metric Information ... 120
System > Input/Output ... 121
CPView Gauges ... 121
CLI ... 121
Metric Information ... 122
System > Gaia ... 123
CPView Gauges ... 123
CLI ... 123
Metric Information ... 123
System > CoreXL ... 127
CPView Gauges ... 127
CLI ... 127
Metric Information ... 128
System > SecureXL ... 129

CPView Gauges ... 130
CLI ... 131
Metric Information ... 131
System > SecureXL > SYN Defender ... 134
CPView Gauges ... 134
CLI ... 134
Metric Information ... 135
System > Flofiler ... 139
CPView Gauges ... 139
Metric Information ... 140
System > Firewall ... 142
CPView Gauges ... 142
CLI ... 142
Metric Information ... 143
System > Process > Top ... 144
CPView Gauges ... 144
CLI ... 144
Metric Information ... 145
System > Process ... 148
CPView Gauges ... 148
CLI ... 148
Metric Information ... 149
ClusterXL ... 160
CPView Gauges ... 160
CLI ... 160
Metric Information ... 160
VSX ... 165
CPView Gauges ... 165
CLI ... 166
Metric Information ... 166

VoIP ... 169
CPView Gauges ... 169
Metric Information ... 169
Network > Heavy Connections ... 172
CPView Gauges ... 172
CLI ... 172
Metric Information ... 173
Network > Network Probes (VPN) ... 178
CLI ... 178
Metric Information ... 179
VPN > Probes ... 182
CLI ... 182
Metric Information ... 183
Blades > Status and Update ... 188
CPView Gauges ... 188
CLI ... 188
Metric Information ... 190
Blades > VPN ... 194
CPView Gauges ... 194
Metric Information ... 195
Blades > IDA ... 210
CPView Gauges ... 210
CLI ... 210
Metric Information ... 211
Hardware > Model ... 219
CPView Gauges ... 219
CLI ... 219
Metric Information ... 219
Hardware > Temperature ... 220
CPView Gauges ... 220

CLI ... 220
Metric Information ... 220
Hardware > Voltage ... 222
CPView Gauges ... 222
CLI ... 222
Metric Information ... 222
Hardware > PSU ... 224
CPView Gauges ... 224
CLI ... 224
Metric Information ... 225
Hardware > Fans ... 226
CPView Gauges ... 226
CLI ... 226
Metric Information ... 226
Hardware > BIOS ... 228
CLI ... 228
Metric Information ... 229
OtlpAgent > Scripts ... 230
Metric Information ... 231
OtlpAgent > Version ... 237
Metric Information ... 237
API on the Management Server ... 238
Metric Information ... 238
Maestro Orchestrator ... 242
CPView Gauges ... 242
CLI ... 242
Metric Information ... 243
Custom Metrics ... 265
Overview ... 265
Procedure ... 265

Notes 270

Introduction

The Check Point CPView service runs on a Check Point server. For information about CPView, see sk101878.

Skyline quickly and efficiently monitors your Security Gateways with industry-standard software and protocols (OpenTelemetry, Prometheus Server, and Grafana Server).

Skyline provides an OpenTelemetry Agent for the Check Point CPView service.

This OpenTelemetry Agent collects and exports health metrics from CPView.

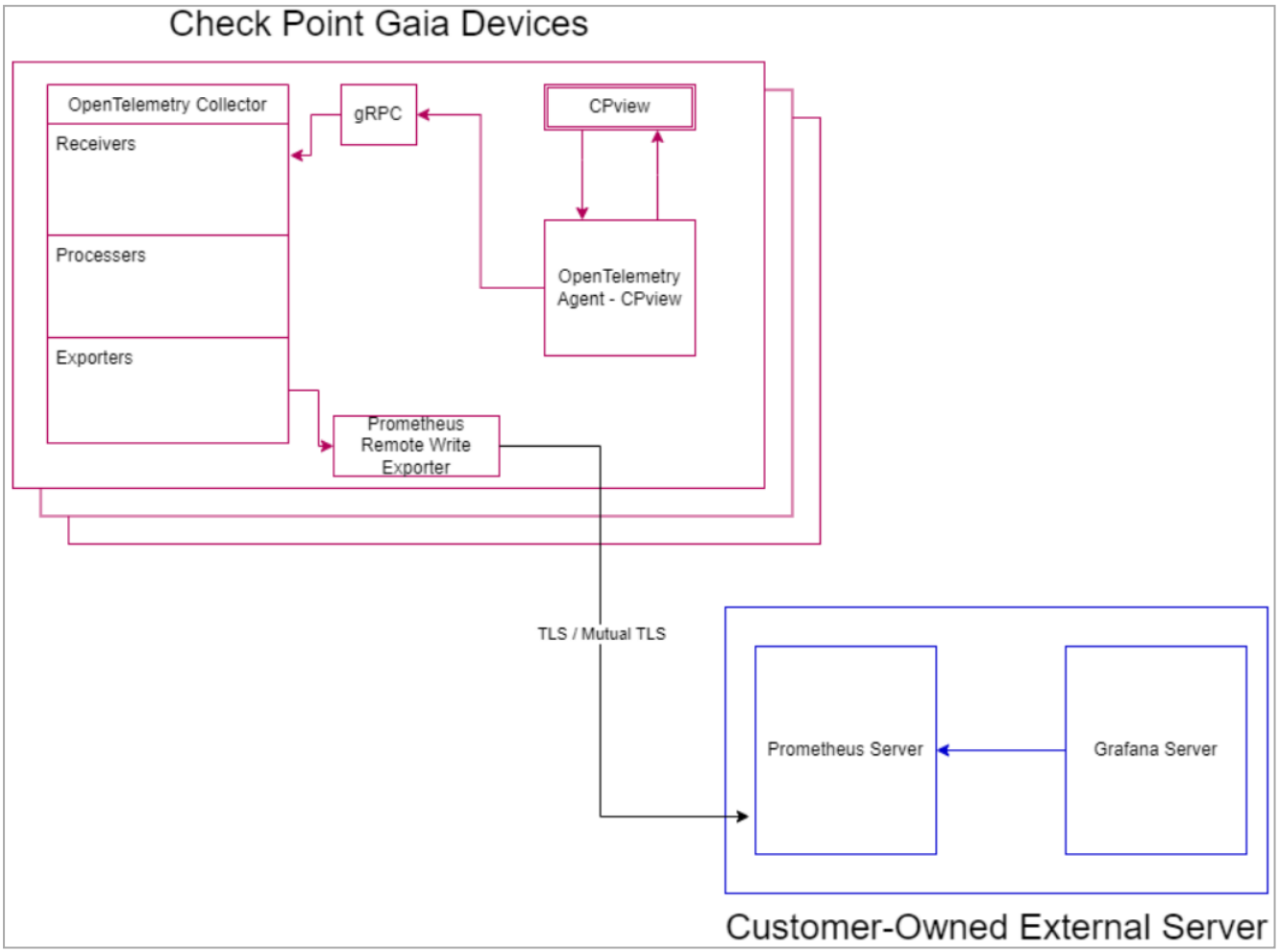
Skyline Architecture

The Skyline architecture includes three primary components:

Component	Description
CPView OpenTelemetry Agent	Runs on Check Point servers. A service that queries CPView at defined intervals, collects the metrics, and exports them to an OpenTelemetry Collector. See sk181615 - OpenTelemetry Agent (OtlpAgent) Release Updates.
OpenTelemetry Collector	Runs on Check Point servers. An open-source service that receives metrics from multiple agents and exports them to an external endpoint (a different OpenTelemetry Collector or a Prometheus Remote-Write). See sk180522 - OpenTelemetry Collector (CPotelcol) Release Updates.
Storage Location	Third-party software that runs on an external server, to which an OpenTelemetry Collector sends the exported metrics. The Prometheus Server receives data from the OpenTelemetry Collector, saves it in a Timeseries Database, and visualizes the data with visualization tools like Grafana. Other third-party monitoring tools are also supported.

In addition, see sk180521 - OpenTelemetry CPviewExporter Release Updates.

Logical Diagram:



Skyline Requirements

See sk178566 > section "**Requirements**".

Skyline Downloads

See sk178566 > section "**Downloads**".

Skyline Known Limitations

See sk178566 > section "**Known Limitations**".

Skyline Configuration

See:

- *"Skyline Configuration on Check Point Servers that run Gaia OS - Prometheus with Grafana" on page 17*
- *"Skyline Configuration on Check Point Servers that run Gaia OS - Other Monitoring Tools" on page 38*
- *"Skyline Configuration on Quantum Spark Appliances" on page 51*

Skyline Troubleshooting and FAQ

See sk179870.

Skyline Metrics

See *"Skyline Metrics Repository" on page 64*.

Skyline Configuration on Check Point Servers that run Gaia OS - Prometheus with Grafana

Note - This section provides the steps for the Prometheus Server and the Grafana Server.
For other monitoring tools, see *"Skyline Configuration on Check Point Servers that run Gaia OS - Other Monitoring Tools" on page 38*.

This section applies to these Check Point Servers:

- Security Gateways / VSX Gateways.
- ClusterXL Members.

  In a Cluster, you must configure all the Cluster Members in the same way.
- Security Groups on Scalable Platforms (ElasticXL Cluster, Maestro, and Scalable Chassis).
- Security Management Servers.
- Multi-Domain Security Management Servers.
- Multi-Domain Log Servers.
- Log Servers.
- SmartEvent Servers.
- Endpoint Security Management Servers.
- Endpoint Policy Servers.

Video Tutorial

https://www.youtube.com/watch?v=FO2Rp9x31i0

Step 1 - Install the Prometheus Server

Note - Skip this step if you have already installed the Prometheus Server

Procedure

Installing a Prometheus Server

To install a Prometheus Server on the external server, refer to the Prometheus installation instructions for the various platforms.

Prometheus Server Default URL

```
http://localhost:9090
```

Configuring a Prometheus Server

1. **Mandatory:** On the Prometheus Server, enable its Remote Write Receiver to get metrics data from the Check Point Servers. Refer to these Prometheus instructions.
2. **Optional:** Use TLS Encryption and Basic authentication to secure the connection between the Prometheus Server and the OpenTelemetry Collector.

   Instructions

   The Prometheus Server and OpenTelemetry Collector support Transport Layer Security (TLS) encryption for their connection. Refer to these Prometheus instructions.

   Check Point also requires you to enable basic authentication to make the security bi-directional, refer to these Prometheus instructions.

   TLS configuration has two main components:

   - A pair of a Key and a Certificate, used to encrypt your communication.
   - (Optional) Certificate Authority (CA) that you trust, used to verify and trust the certificate of the other endpoint with which you communicate. If the certificate of the other endpoint is unknown to the CA, the communication is rejected.

   You can create these certificates:

   - CA-signed certificates: You create a key and a certificate request, which is then signed by the CA.

- Self-signed certificates: You create a key and a certificate that is signed by the user.

The steps below describe a self-signed certificate.

To configure TLS, you must create two pairs of a key and a certificate on the Prometheus Server:

a. Create a self-signed certificate and a private key on the Prometheus Server:

   i. Create the file called `openssl.conf` with the template below.

   Enter the applicable information in the "`[ dn ]`" and "`[ alt_names ]`" sections.

   Important - The Prometheus Server and the OpenTelemetry Collector must have different hostnames and IP addresses.

   Template:

```
[ req ]
default_bits       = 4096
default_md         = sha256
req_extensions     = v3_req
distinguished_name = dn
prompt = no

[ v3_req ]
subjectAltName = @alt_names

[ dn ]
C = <Country Name>
ST = <State Or Province Name>
L = <Locality>
O = <Organization>
OU = <Organization Unit>
CN = <Common Name>

[ alt_names ]
DNS = <HOSTNAME>
IP = <IP ADDRESS>
```

ii. Generate the key and certificate pair on any Gaia OS server (in the Expert mode):

```
cpopenssl req -x509 -newkey rsa:4096 -nodes -
config openssl.conf -keyout mykey.key -out
mycert.crt -extensions v3_req
```

This command creates two files in the current working directory:

- `mykey.key`
- `mycert.crt`

Move these two files from the Gaia OS server to the Prometheus Server.

b. Use the newly generated key and certificate file to configure TLS on the Prometheus Server in the `web-config.yaml` configuration file (**you may need to create this file**).

Example of a `web-config.yaml` file:

```
tls_server_config:
   key_file: /home/admin/mykey.key
   cert_file: /home/admin/mycert.crt
```

Step 2 - Install the Grafana Server

Note - Skip this step if you have already installed the Grafana Server.

Procedure

Installing a Grafana Server

To install a Grafana Server on the external server, refer to the Grafana installation instructions.

Note - You can install the Grafana Server on the same server that contains your Prometheus Server instance.

Grafana Server Default URL

```
http://localhost:3000
```

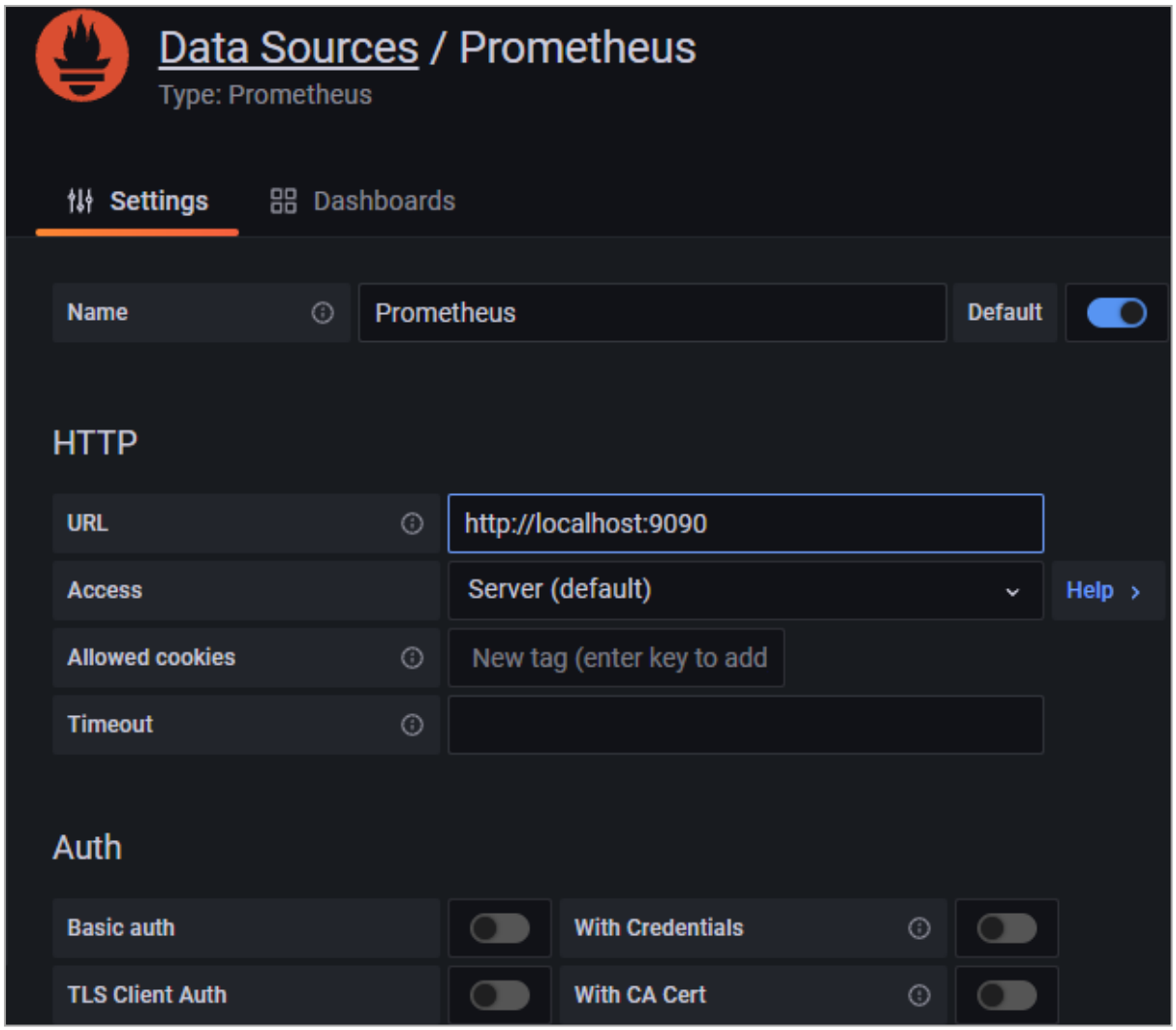
Configuring a Grafana Server (connecting Data-source and Dashboards)

1. Log in to Grafana and add a new Prometheus Data Source with the Prometheus Server you created.

   On the side panel, click **Configuration** > **Data sources** > click the **Add data source** button > select the **Prometheus** data source.

   Make sure to mark the Prometheus data source as **Default**.

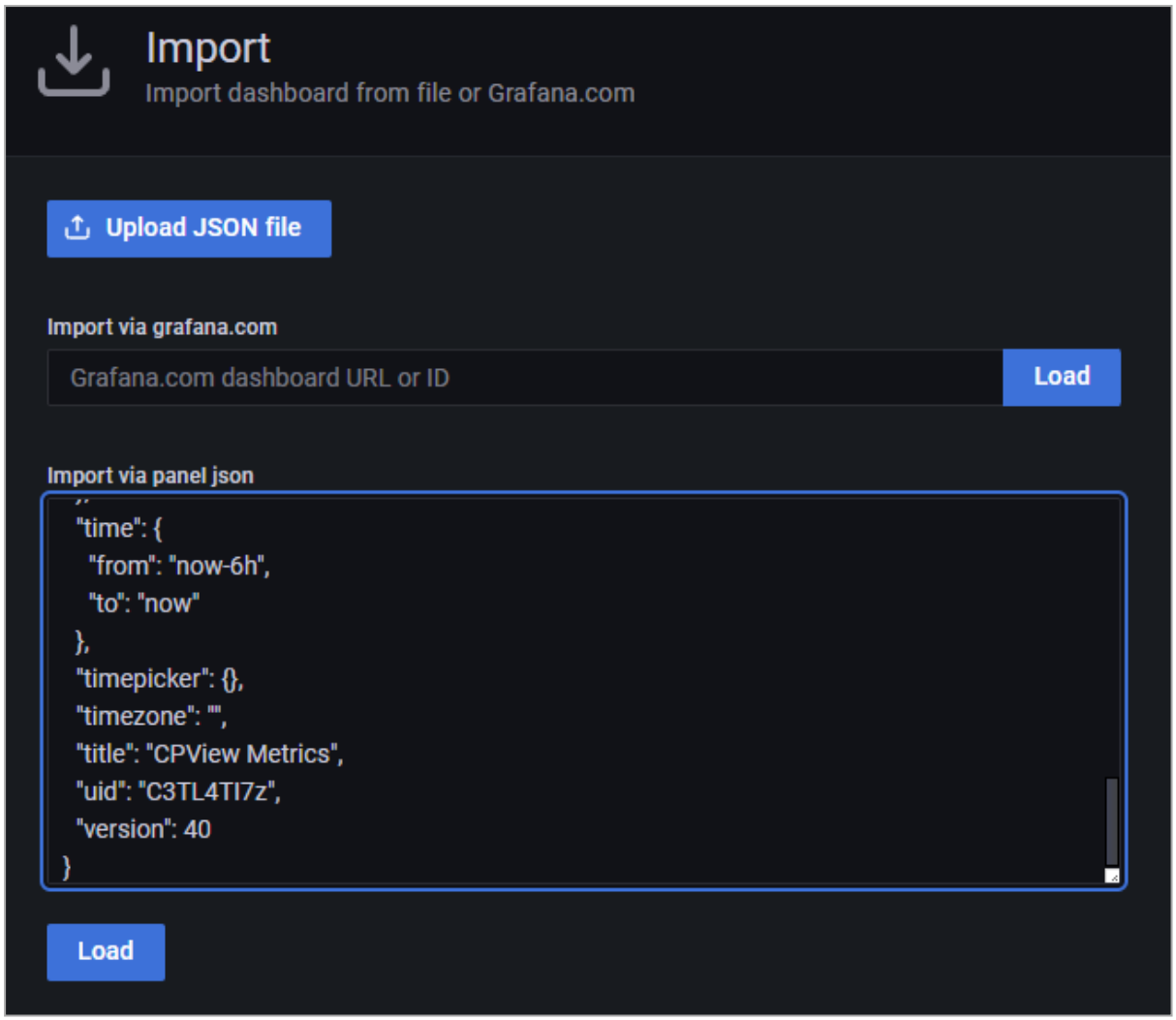
Example:



2. Import the Check Point Grafana dashboard to start monitoring your environment.

   On the side panel, click **Create** > **Import** > upload the applicable JSON file for a dashboard.

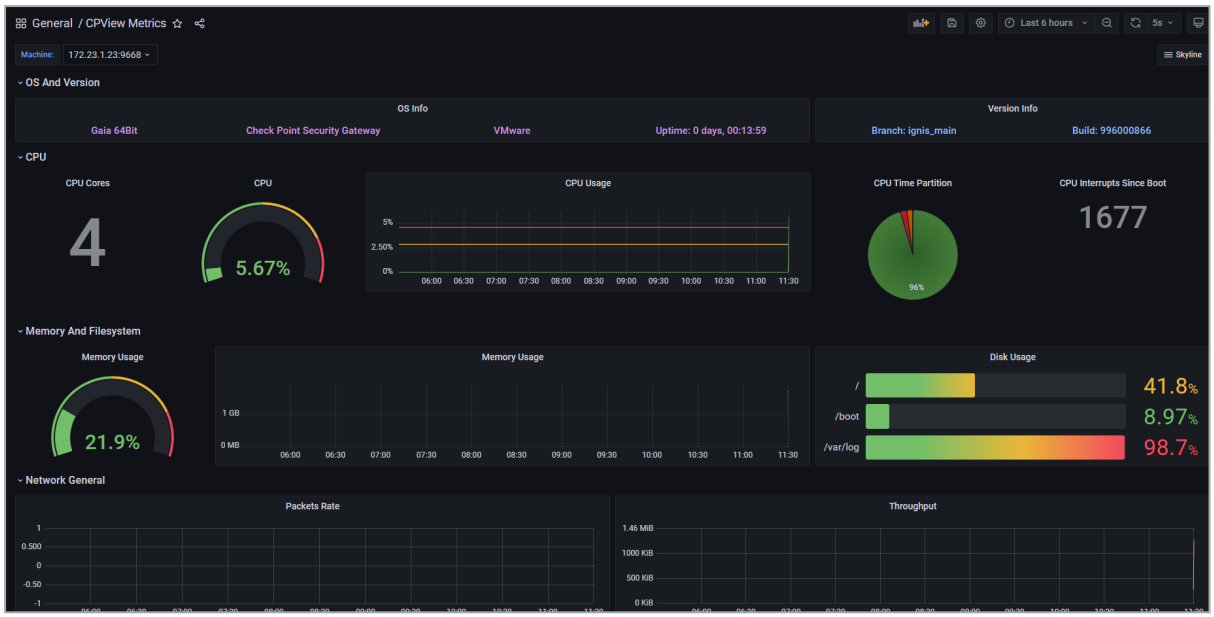
Example:



3. You can modify the existing dashboards or create new dashboards according to your needs.

   To see a full description of all the data exposed by Skyline, see *"Skyline Metrics Repository" on page 64*.

Example:


General / CPView Metrics
Last 6 hours
5s
Machine:
172.23.1.23:9668
Skyline
OS And Version
OS Info
Gaia 64Bit
Check Point Security Gateway
VMware
Uptime: 0 days, 00:13:59
Version Info
Branch: ignis_main
Build: 996000866
CPU
CPU Cores
4
CPU
5.67%
CPU Usage
5%
2.50%
0%
06:00 06:30 07:00 07:30 08:00 08:30 09:00 09:30 10:00 10:30 11:00 11:30
CPU Time Partition
96%
CPU Interrupts Since Boot
1677
Memory And Filesystem
Memory Usage
21.9%
Memory Usage
1 GB
0 MB
06:00 06:30 07:00 07:30 08:00 08:30 09:00 09:30 10:00 10:30 11:00 11:30
Disk Usage
/
/boot
/var/log
41.8%
8.97%
98.7%
Network General
Packets Rate
1
0.500
0
-0.50
-1
Throughput
1.46 MiB
1000 KiB
500 KiB
0 KiB

Step 3 - Install the OpenTelemetry Agent and OpenTelemetry Collector on the Check Point Server

Procedure

On an online Check Point Server:

The OpenTelemetry Agent (`OtlpAgent`) and the OpenTelemetry Collector (`CPotelcol`) packages are installed automatically on all applicable Check Point Servers, if you allowed the Automatic Update downloads as described in sk94508.

On an offline Check Point Server:

The minimum required OpenTelemetry Agent and OpenTelemetry Collector packages are also installed as part of the Jumbo Hotfix Accumulator installation.

See sk178566 > section "**Requirements**".

Related offline update packages:

- sk180522 - CPotelcol (OpenTelemetry Collector) Release Updates
- sk180521 - CPviewExporter Release Updates

Step 4 - Configure the OpenTelemetry Collector on the Check Point Server to work with the Prometheus Server

Important Notes

Important:

- Do not cut and paste the commands below. Enter them manually.
- In a Cluster, you must configure all the Cluster Members in the same way.
- In Management High Availability, we recommend to configure all the Management Servers in the same way.
- On a Maestro Orchestrator (MHO):
  You can run the CLI command "`sklnctl`" (in the Expert mode) or run the Gaia REST API command (requires Gaia API v1.7 or higher).
  The CLI command runs on the Orchestrator and configures only the Orchestrator.
- In a Scalable Platform Security Group:
  You can run the CLI command "`g_all sklnctl`" (in the Expert mode) or run the Gaia REST API command (requires Gaia API v1.7 or higher).
- For the applicable Gaia REST API commands, refer to the *Check Point Gaia API Reference* (v1.7 and higher):
  - In Gaia API v1.8 and higher: Section "*Diagnostics*" > Section "*OpenTelemetry*".
  - In Gaia API v1.7: Section "*OpenTelemetry*".

This step provides two different procedures - for an on-premises Prometheus Server, and for a Prometheus Server in AWS.

Procedure to work with an on-premises Prometheus Server

1. Prepare the required payload for the command:

   Notes:
   - Download the sample payload files from sk178566 > section "**Downloads**".
     Replace the placeholder strings "`<...>`" in the sample files with your actual strings.
   - In the "`basic`" section, configure the "`username`" and "`password`" attributes to your monitoring server's username / password.
     The "`password`" attribute supports only these characters:
     - uppercase letters (`A-Z`)
     - lowercase letters (`a-z`)
     - digits (`0-9`)
     - underscore (`_`)
     - comma (`,`)
     - period (`.`)
     - backslash (`\`)
     - slash (`/`)
     - hyphen (`-`)
   - In the "`ca-public-key`" section, configure the TLS settings with the CA certificate of your monitoring server (as PEM X509), or use the self-signed certificate you generated previously.
     Replace "`<CERTIFICATE>`" with the CA certificate of the monitoring server (PEX X509) - paste the entire string:

     ```
     -----BEGIN CERTIFICATE-----<BASE64_TEXT>-----END CERTIFICATE-----
     ```

   - In the "`url`" attribute, configure your monitoring server's IP address / URL.
     If you do not wish to use TLS encryption in labs or test environments, then make these changes in the payload:
     a. In the "`url`" attribute, change the URL protocol from "`https://`" to "`http://`".
     b. Remove the "`client-auth`" and "`server-auth`" attributes.
        Note - When you apply the payload, this warning appears: "`it is recommended to have both client and server authentication`").

Example JSON payloads:

Example payload - connection with TLS

```
{
    "enabled": true,
    "export-targets": {
        "add": [
            {
                "client-auth": {
                    "basic": {
                        "username": "<USERNAME>",
                        "password": "<PASSWORD>"
                    }
                },
                "enabled": true,
                "server-auth": {
                    "ca-public-key": {
                        "type": "PEM-X509",
                        "value": "<CERTIFICATE>"
                    }
                },
                "type": "prometheus-remote-write",
                "url": "https://<EXTERNAL_PROMETHEUS_IP_ADDRESS>:9090/api/v1/write"
            }
        ]
    }
}
```

Example payload - connection without TLS

```
{
    "enabled": true,
    "export-targets": {
        "add": [
            {
                "enabled": true,
                "type": "prometheus-remote-write",
                "url": "http://<EXTERNAL_PROMETHEUS_IP_ADDRESS>:9090/api/v1/write"
            }
        ]
    }
}
```

2. Run the configuration command to apply the payload - either the CLI command (in the Expert mode), or the Gaia REST API command:

- Method 1 - Run the CLI command "`sklnctl`":

  a. Save the JSON payload in a file (for example, `/home/admin/payload.json`).

  b. On a Scalable Platform Security Group, copy the JSON payload file to all Security Group Members:

  ```
  asg_cp2blades /home/admin/payload.json
  ```

  c. Run this command:

  - On a Security Gateway / each Cluster Member / Management Server / Log Server / SmartEvent Server:

  ```
  sklnctl export --set "$(cat
  /home/admin/payload.json)"
  ```

  - On a Scalable Platform Security Group:

  ```
  g_all sklnctl export --set "$(cat
  /home/admin/payload.json)"
  ```

- Method 2 - Run the Gaia REST API command "`set-open-telemetry`" (requires Gaia API v1.7 and higher):

```
POST{server}/v1.7/set-open-telemetry
Content-Type: application/json
X-chkp-sid: {{session}}
{
   <JSON Payload>
}
```

Note - To disable Skyline completely:

a. Change the value of the "`enabled`" attribute in the JSON payload file:
from `"enabled": true`
to `"enabled": false`

b. Run the API command again

Procedure to work with a Prometheus Server in AWS

1. Prepare the required payload for the command:

   Notes:
   - Refer to the AWS official documentation for detailed information about how to generate the keys.
   - The Session Token is optional.

   To work with a single Export Target over HTTPS:

```
{
    "enabled": true,
    "export-targets": {
        "add": [
            {
                "server-auth": {
                    "sigv4auth": {
                        "region": "<Region>",
                        "aws-access-key-id": "<Access Key
ID>",
                        "aws-secret-access-key": "<Access
Key>",
                        "session-token": "<Session Token>"
                    }
                },
                "enabled": true,
                "type": "prometheus-remote-write",
                "url": "https://<PROMETHEUS_IP_
ADDRESS>:9090/api/v1/write"
            }
        ]
    }
}
```

To work with multiple Export Targets over HTTPS (this example shows two):

```
{
    "enabled": true,
    "export-targets": {
        "add": [
            {
                "server-auth": {
                    "sigv4auth": {
                        "region": "<Region>",
                        "aws-access-key-id": "<Access Key
ID>",
                        "aws-secret-access-key": "<Access
Key>",
                        "session-token": "<Session Token>"
                    }
                },
                "enabled": true,
                "type": "prometheus-remote-write-1",
                "url": "https://<PROMETHEUS_IP_ADDRESS_
#1>:9090/api/v1/write",
                "name": "My-Prometheus-Target-1"
            },
            {
                "enabled": true,
                "type": "prometheus-remote-write-2",
                "url": "https://<PROMETHEUS_IP_ADDRESS_
#2>:9090/api/v1/write",
                "name": "My-Prometheus-Target-2"
            }
        ]
    }
}
```

To work with multiple Export Targets over HTTPS and over HTTP (this example shows two):

```
{
    "enabled": true,
    "export-targets": {
        "add": [
            {
                "server-auth": {
                    "sigv4auth": {
                        "region": "<Region>",
                        "aws-access-key-id": "<Access Key
ID>",
                        "aws-secret-access-key": "<Access
Key>",
                        "session-token": "<Session Token>",
                        "enabled": true,
                        "type": "prometheus-remote-write-1",
                        "url": "https://<PROMETHEUS_IP_
ADDRESS_#1>:9090/api/v1/write",
                        "name": "My-Prometheus-Target-1"
                    }
                }
            },
            {
                "enabled": true,
                "type": "prometheus-remote-write-2",
                "url": "http://<PROMETHEUS_IP_ADDRESS_
#2>:9090/api/v1/write",
                "name": "My-Prometheus-Target-2"
            }
        ]
    }
}
```

2. Run the configuration command to apply the payload - either the CLI command (in the Expert mode), or the Gaia REST API command:

   - Method 1 - Run the CLI command "`sklnctl`":

     a. Save the JSON payload in a file (for example, `/home/admin/payload_AWS.json`).

     b. On a Scalable Platform Security Group, copy the JSON payload file to all Security Group Members:

     ```
     asg_cp2blades /home/admin/payload_AWS.json
     ```

     c. Run this command:

     - On a Security Gateway / each Cluster Member / Management Server / Log Server / SmartEvent Server:

     ```
     sklnctl export --set "$(cat
     /home/admin/payload_AWS.json)"
     ```

     - On a Scalable Platform Security Group:

     ```
     g_all sklnctl export --set "$(cat
     /home/admin/payload_AWS.json)"
     ```

   - Method 2 - Run the Gaia REST API command "`set-open-telemetry`" (requires Gaia API v1.7 and higher):

     ```
     POST{server}/v1.7/set-open-telemetry
     Content-Type: application/json
     X-chkp-sid: {{session}}
     {
        <JSON AWS Payload>
     }
     ```

Step 5 - Configure the filter for the OpenTelemetry Collector exported metrics

Procedure

The `sklnctl` tool configures the OpenTelemetry Collector.

The OpenTelemetry Collector filter works on the allow-list basis.

Note - On a Scalable Platform Security Group, run the "`g_all sklnctl <options>`" command.

These are the available commands (in the Expert mode):

- To show the currently exported metrics:

```
sklnctl otelcol metrics --show
```

- To check if default metrics are exported:

```
sklnctl otelcol metrics --is-default
```

- To add metrics to the allow-list:

```
sklnctl otelcol metrics --add <metric-id1> <metric-id2>
<metric-id3> ...
```

- To remove metrics from the allow-list:

```
sklnctl otelcol metrics --remove <metric-id1> <metric-id2>
<metric-id3> ...
```

- To reset the allow-list to the default:

```
sklnctl otelcol metrics --reset
```

Example workflow (in the Expert mode):

1. Show the currently exported metrics:

   ```
   sklnctl otelcol metrics --show > /var/log/metrics.txt
   ```

2. Edit the file to keep only the desired metrics:

   ```
   vi /var/log/metrics.txt
   ```

   See *"Skyline Metrics Repository" on page 64*.

3. On a Scalable Platform Security Group, copy the file to all Security Group Members:

   ```
   asg_cp2blades /var/log/metrics.txt
   ```

4. Add the desired metrics to the allow-list:

   - On a Security Gateway / each Cluster Member / Management Server / Log Server / SmartEvent Server:

     ```
     sklnctl otelcol metrics --add $(cat
     /var/log/metrics.txt | tr '\n' ' ')
     ```

   - On a Scalable Platform Security Group:

     ```
     g_all sklnctl otelcol metrics --add $(cat
     /var/log/metrics.txt | tr '\n' ' ')
     ```

Step 6 - Configure Access Control Policy

If you configured Skyline on a Security Gateway, ClusterXL, or Scalable Platform Security Group, then you must make sure your Access Control Policy allows the connection to the Prometheus Server to send the exported metrics.

You must configure the required rule on the Management Server (in SmartConsole or with Management API) and install the policy.

See the:

- *Quantum Security Management Administration Guide* for your version.
- *Check Point Management API Reference*.

Skyline Configuration on Check Point Servers that run Gaia OS - Other Monitoring Tools

Best Practice - Use the Prometheus Server with the Grafana Server.
See *"Skyline Configuration on Check Point Servers that run Gaia OS - Prometheus with Grafana" on page 17*.

Skyline supports other third-party monitoring tools (to configure these tools, refer to the third-party official documentation):

- Splunk
- SolarWinds
- Dynatrace
- VictoriaMetrics
- Datadog

This section applies to these Check Point Servers:

- Security Gateways / VSX Gateways.
- ClusterXL Members.

  In a Cluster, you must configure all the Cluster Members in the same way.
- Security Groups on Scalable Platforms (ElasticXL Cluster, Maestro, and Scalable Chassis).
- Security Management Servers.
- Multi-Domain Security Management Servers.
- Multi-Domain Log Servers.
- Log Servers.
- SmartEvent Servers.
- Endpoint Security Management Servers.
- Endpoint Policy Servers.

Step 1 - Install the Third-Party Monitoring Tool

Refer to the third-party official documentation.

Step 2 - Install the OpenTelemetry Agent and OpenTelemetry Collector on the Check Point Server

Procedure

On an online Check Point Server:

The OpenTelemetry Agent (`OtlpAgent`) and the OpenTelemetry Collector (`CPotelcol`) packages are installed automatically on all applicable Check Point Servers, if you allowed the Automatic Update downloads as described in sk94508.

On an offline Check Point Server:

The minimum required OpenTelemetry Agent and OpenTelemetry Collector packages are also installed as part of the Jumbo Hotfix Accumulator installation.

See sk178566 > section "**Requirements**".

Related offline update packages:

- sk181615 - OpenTelemetry Agent (OtlpAgent) Release Updates
- sk180522 - OpenTelemetry Collector (CPotelcol) Release Updates
- sk180521 - OpenTelemetry CPviewExporter Release Updates

Step 3 - Configure the OpenTelemetry Collector on the Check Point Server to work with the Third-Party Monitoring Tool

Important Notes

Important:

- Do not cut and paste the commands below. Enter them manually.
- In a Cluster, you must configure all the Cluster Members in the same way.
- In Management High Availability, we recommend to configure all the Management Servers in the same way.
- On a Maestro Orchestrator (MHO):
  You can run the CLI command "`sklnctl`" (in the Expert mode) or run the Gaia REST API command (requires Gaia API v1.7 or higher).
  The CLI command runs on the Orchestrator and configures only the Orchestrator.
- In a Scalable Platform Security Group:
  You can run the CLI command "`g_all sklnctl`" (in the Expert mode) or run the Gaia REST API command (requires Gaia API v1.7 or higher).
- For the applicable Gaia REST API commands, refer to the *Check Point Gaia API Reference* (v1.7 and higher):
  - In Gaia API v1.8 and higher: Section "*Diagnostics*" > Section "*OpenTelemetry*".
  - In Gaia API v1.7: Section "*OpenTelemetry*".

Procedure

1. Prepare the required payload for the command:

   Notes:

   - Replace the placeholder strings "<...>" below with your actual strings.
   - The "name" attribute is optional, but recommended.

     Must contain only lowercase Latin letters (a-z).
   - To configure a TLS connection between the OpenTelemetry Collector and monitoring server:

     In the "ca-public-key" section, configure the TLS settings with the CA certificate of your monitoring server (as PEM X509), or use a self-signed certificate you generate for yourself.

     In the example payloads below, replace "<SERVER-CERTIFICATE>" with the CA certificate of the monitoring server (PEX X509) - paste the entire string:

     ```
     -----BEGIN CERTIFICATE-----<BASE64_TEXT>-----END
     CERTIFICATE-----
     ```
   - In the "url" attribute, configure your monitoring server's IP address / URL.

     If you do not wish to use TLS encryption in labs or test environments, then make these changes in the payload:

     a. In the "url" attribute, change the URL protocol from "https://" to "http://".

     b. Remove the "client-auth" and "server-auth" attributes.

        Note - When you apply the payload, this warning appears: "it is recommended to have both client and server authentication").

Example JSON payloads for Splunk:

Note - For information about a HEC Token for Splunk, click here.

JSON payload for Splunk - connection with TLS

```
{
    "enabled": true,
    "export-targets": {
        "add": [
            {
                "client-auth": {
                    "token": {
                        "custom-header": {
                            "key": "token",
                            "value": "<YOUR_HEC_TOKEN>"
                        }
                    }
                },
                "server-auth": {
                    "ca-public-key": {
                        "type": "PEM-X509",
                        "value": "<SERVER-CERTIFICATE>"
                    }
                },
                "enabled": true,
                "type": "splunk_hec",
                "name": "splunk-tls-01",
                "url": "https://<FQDN_or_IP_of_SPLUNK_
SERVER>:8088/services/collector/event"
            }
        ]
    }
}
```

JSON payload for Splunk - connection without TLS

```
{
    "enabled": true,
    "export-targets": {
        "add": [
            {
                "client-auth": {
                    "token": {
                        "custom-header": {
                            "key": "token",
                            "value": "<YOUR_HEC_TOKEN>"
                        }
                    }
                },
                "enabled": true,
                "type": "splunk_hec",
                "name": "splunk-no-tls-01",
                "url": "https://<FQDN_or_IP_of_SPLUNK_
SERVER>:8088/services/collector/event"
            }
        ]
    }
}
```

Example JSON payloads for SolarWinds:

Note - For information about an API Token for SolarWinds, click here.

JSON payload for SolarWinds - connection with TLS

```
{
    "enabled": true,
    "export-targets": {
        "add": [
            {
                "client-auth": {
                    "token": {
                        "custom-header": {
                            "key": "Authorization",
                            "value": "Bearer <YOUR_API_TOKEN>"
                        }
                    }
                },
                "server-auth": {
                    "ca-public-key": {
                        "type": "PEM-X509",
                        "value": "<SERVER-CERTIFICATE>"
                    }
                },
                "enabled": true,
                "type": "otlp",
                "name": "solarwinds-tls-01",
                "url": "otel.collector.<YOUR_REGION>.cloud.solarwinds.com:443"
            }
        ]
    }
}
```

JSON payload for SolarWinds - connection without TLS

```
{
    "enabled": true,
    "export-targets": {
        "add": [
            {
                "client-auth": {
                    "token": {
                        "custom-header": {
                            "key": "Authorization",
                            "value": "Bearer <YOUR_API_TOKEN>"
                        }
                    }
                },
                "enabled": true,
                "type": "otlp",
                "name": "solarwinds-no-tls-01",
                "url": "otel.collector.<YOUR_REGION>.cloud.solarwinds.com:443"
            }
        ]
    }
}
```

Example JSON payloads for Dynatrace:

Note - For information about a Access Token for Dynatrace, click here.

JSON payload for Dynatrace - connection with TLS

```
{
    "enabled": true,
    "export-targets": {
        "add": [
            {
                "client-auth": {
                    "token": {
                        "custom-header": {
                            "key": "Authorization",
                            "value": "Api-Token <YOUR_ACCESS_TOKEN>"
                        }
                    }
                },
                "server-auth": {
                    "ca-public-key": {
                        "type": "PEM-X509",
                        "value": "<SERVER-CERTIFICATE>"
                    }
                },
                "enabled": true,
                "type": "otlphttp",
                "name": "dynatrace-tls-01",
                "url": "https://<FQDN_or_IP_of_DYNATRACE_
SERVER>.live.dynatrace.com/api/v2/otlp"
            }
        ]
    }
}
```

JSON payload for Dynatrace - connection without TLS

```
{
    "enabled": true,
    "export-targets": {
        "add": [
            {
                "client-auth": {
                    "token": {
                        "custom-header": {
                            "key": "Authorization",
                            "value": "Api-Token <YOUR_ACCESS_TOKEN>"
                        }
                    }
                },
                "enabled": true,
                "type": "otlphttp",
                "name": "dynatrace-no-tls-01",
                "url": "https://<FQDN_or_IP_of_DYNATRACE_
SERVER>.live.dynatrace.com/api/v2/otlp"
            }
        ]
    }
}
```

Example JSON payloads for VictoriaMetrics:

JSON payload for VictoriaMetrics - connection with TLS

Note - In the "`basic`" section, configure the "`username`" and "`password`" attributes to your monitoring server's username / password.
The "`password`" attribute supports only these characters:

- uppercase letters (`A-Z`)
- lowercase letters (`a-z`)
- digits (`0-9`)
- underscore (`_`)
- comma (`,`)
- period (`.`)
- backslash (`\`)
- slash (`/`)
- hyphen (`-`)

```
{
    "enabled": true,
    "export-targets": {
        "add": [
            {
                "client-auth": {
                    "basic": {
                        "username": "<USERNAME_on_VICTORIAMETRICS_SERVER>",
                        "password": "<PASSWORD_on_VICTORIAMETRICS_SERVER>"
                    }
                },
                "enabled": true,
                "server-auth": {
                    "ca-public-key": {
                        "type": "PEM-X509",
                        "value": "<SERVER-CERTIFICATE>"
                    }
                },
                "type": "prometheus-remote-write",
                "url": "https://<EXTERNAL_IP_ADDRESS_of_VICTORIAMETRICS_
SERVER>:9090/api/v1/write"
            }
        ]
    }
}
```

JSON payload for VictoriaMetrics - connection without TLS

```
{
    "enabled": true,
    "export-targets": {
        "add": [
            {
                "enabled": true,
                "type": "prometheusremotewrite",
                "url": "https://<EXTERNAL_IP_ADDRESS_of_VICTORIAMETRICS_
SERVER>:8088/services/collector"
            }
        ]
    }
}
```

Example JSON payloads for Datadog:

Warning - Skyline does **not** support a JSON payload that contains other export targets together with the Datadog export target.

JSON payload for Datadog - connection with TLS

```
{
    "enabled": true,
    "export-targets": {
        "add": [
            {
                "client-auth": {
                    "token": {
                        "custom-header": {
                            "key": "apikey",
                            "value": "<YOUR-API-KEY>"
                        }
                    }
                },
                "server-auth": {
                    "ca-public-key": {
                        "type": "PEM-X509",
                        "value": "<SERVER-CERTIFICATE>"
                    }
                },
                "enabled": true,
                "type": "datadog",
                "name": "datadog-tls-01",
                "url": "https://<URL_of_YOUR_DATADOG_INSTANCE>"
            }
        ]
    }
}
```

JSON payload for Datadog - connection without TLS

```
{
    "enabled": true,
    "export-targets": {
        "add": [
            {
                "client-auth": {
                    "token": {
                        "custom-header": {
                            "key": "apikey",
                            "value": "<YOUR-API-KEY>"
                        }
                    }
                },
                "enabled": true,
                "type": "datadog",
                "name": "datadog-no-tls-01",
                "url": "https://<URL_of_YOUR_DATADOG_INSTANCE>"
            }
        ]
    }
}
```

2. Run the configuration command to apply the JSON payload - either the CLI command (in the Expert mode), or the Gaia REST API command:

   - Method 1 - Run the CLI command "`sklnctl`":

     a. Save the JSON payload in a file (for example, `/home/admin/payload.json`).

     b. On a Scalable Platform Security Group, copy the JSON payload file to all Security Group Members:

     ```
     asg_cp2blades /home/admin/payload.json
     ```

     c. Run this command:

     - On a Security Gateway / each Cluster Member / Management Server / Log Server / SmartEvent Server:

     ```
     sklnctl export --set "$(cat
     /home/admin/payload.json)"
     ```

     - On a Scalable Platform Security Group:

     ```
     g_all sklnctl export --set "$(cat
     /home/admin/payload.json)"
     ```

   - Method 2 - Run the Gaia REST API command "`set-open-telemetry`" (requires Gaia API v1.7 and higher):

     ```
     POST{server}/v1.7/set-open-telemetry
     Content-Type: application/json
     X-chkp-sid: {{session}}
     {
        <JSON Payload>
     }
     ```

     Note - To disable Skyline completely:

     a. Change the value of the "`enabled`" attribute in the JSON payload file:
     from `"enabled": true`
     to `"enabled": false`

     b. Run the API command again

Step 4 - Configure the filter for the OpenTelemetry Collector exported metrics

Procedure

The `sklnctl` tool configures the OpenTelemetry Collector.

The OpenTelemetry Collector filter works on the allow-list basis.

Note - On a Scalable Platform Security Group, run the "`g_all sklnctl <options>`" command.

These are the available commands (in the Expert mode):

- To show the currently exported metrics:

```
sklnctl otelcol metrics --show
```

- To check if default metrics are exported:

```
sklnctl otelcol metrics --is-default
```

- To add metrics to the allow-list:

```
sklnctl otelcol metrics --add <metric-id1> <metric-id2>
<metric-id3> ...
```

- To remove metrics from the allow-list:

```
sklnctl otelcol metrics --remove <metric-id1> <metric-id2>
<metric-id3> ...
```

- To reset the allow-list to the default:

```
sklnctl otelcol metrics --reset
```

Example workflow (in the Expert mode):

1. Show the currently exported metrics:

   ```
   sklnctl otelcol metrics --show > /var/log/metrics.txt
   ```

2. Edit the file to keep only the desired metrics:

   ```
   vi /var/log/metrics.txt
   ```

   See *"Skyline Metrics Repository" on page 64*.

3. On a Scalable Platform Security Group, copy the file to all Security Group Members:

   ```
   asg_cp2blades /var/log/metrics.txt
   ```

4. Add the desired metrics to the allow-list:

   - On a Security Gateway / each Cluster Member / Management Server / Log Server / SmartEvent Server:

     ```
     sklnctl otelcol metrics --add $(cat
     /var/log/metrics.txt | tr '\n' ' ')
     ```

   - On a Scalable Platform Security Group:

     ```
     g_all sklnctl otelcol metrics --add $(cat
     /var/log/metrics.txt | tr '\n' ' ')
     ```

Step 5 - Configure Access Control Policy

If you configured Skyline on a Security Gateway, ClusterXL, or Scalable Platform Security Group, then you must make sure your Access Control Policy allows the connection to the Third-Party Monitoring Tool to send the exported metrics.

You must configure the required rule on the Management Server (in SmartConsole or with Management API) and install the policy.

See the:

- *Quantum Security Management Administration Guide* for your version.
- *Check Point Management API Reference*.

Skyline Configuration on Quantum Spark Appliances

Video Tutorial

https://www.youtube.com/watch?v=FO2Rp9x31i0

Step 1 - Install the Prometheus Server

Note - Skip this step if you have already installed the Prometheus Server

Procedure

Installing a Prometheus Server

To install a Prometheus Server on the external server, refer to the Prometheus installation instructions for the various platforms.

Prometheus Server Default URL

```
http://localhost:9090
```

Configuring a Prometheus Server

1. **Mandatory:** On the Prometheus Server, enable its Remote Write Receiver to get metrics data from the Check Point Servers. Refer to these Prometheus instructions.
2. **Optional:** Use TLS Encryption and Basic authentication to secure the connection between the Prometheus Server and the OpenTelemetry Collector.

   Instructions

   The Prometheus Server and OpenTelemetry Collector support Transport Layer Security (TLS) encryption for their connection. Refer to these Prometheus instructions.

   Check Point also requires you to enable basic authentication to make the security bi-directional, refer to these Prometheus instructions.

   TLS configuration has two main components:

- A pair of a Key and a Certificate, used to encrypt your communication.
- (Optional) Certificate Authority (CA) that you trust, used to verify and trust the certificate of the other endpoint with which you communicate. If the certificate of the other endpoint is unknown to the CA, the communication is rejected.

You can create these certificates:

- CA-signed certificates: You create a key and a certificate request, which is then signed by the CA.
- Self-signed certificates: You create a key and a certificate that is signed by the user.

The steps below describe a self-signed certificate.

To configure TLS, you must create two pairs of a key and a certificate on the Prometheus Server:

a. Create a self-signed certificate and a private key on the Prometheus Server:

   i. Create the file called `openssl.conf` with the template below.

      Enter the applicable information in the "`[ dn ]`" and "`[ alt_names ]`" sections.

      Important - The Prometheus Server and the OpenTelemetry Collector must have different hostnames and IP addresses.

Template:

```
[ req ]
default_bits        = 4096
default_md          = sha256
req_extensions      = v3_req
distinguished_name = dn
prompt = no

[ v3_req ]
subjectAltName = @alt_names

[ dn ]
C = <Country Name>
ST = <State Or Province Name>
L = <Locality>
O = <Organization>
OU = <Organization Unit>
CN = <Common Name>

[ alt_names ]
DNS = <HOSTNAME>
IP = <IP ADDRESS>
```

ii. Generate the key and certificate pair on any Gaia OS server (in the Expert mode):

```
cpopenssl req -x509 -newkey rsa:4096 -nodes -
config openssl.conf -keyout mykey.key -out
mycert.crt -extensions v3_req
```

This command creates two files in the current working directory:

- `mykey.key`
- `mycert.crt`

Move these two files from the Gaia OS server to the Prometheus Server.

b. Use the newly generated key and certificate file to configure TLS on the Prometheus Server in the `web-config.yaml` configuration file (**you may need to create this file**).

Example of a `web-config.yaml` file:

```
tls_server_config:
   key_file: /home/admin/mykey.key
   cert_file: /home/admin/mycert.crt
```

Step 2 - Install the Grafana Server

Note - Skip this step if you have already installed the Grafana Server.

Procedure

Installing a Grafana Server

To install a Grafana Server on the external server, refer to the Grafana installation instructions.

Note - You can install the Grafana Server on the same server that contains your Prometheus Server instance.

Grafana Server Default URL

```
http://localhost:3000
```

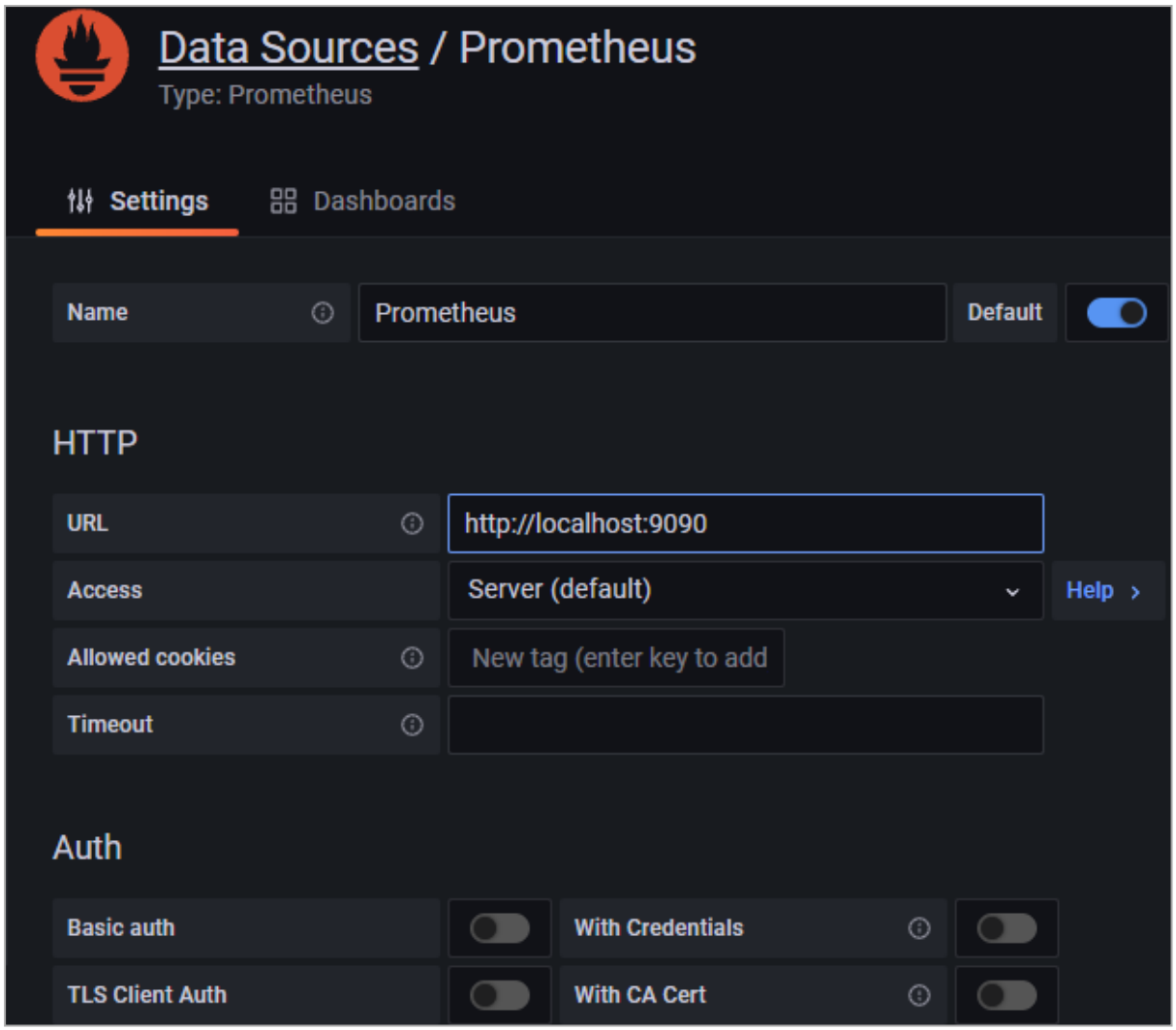
Configuring a Grafana Server (connecting Data-source and Dashboards)

1. Log in to Grafana and add a new Prometheus Data Source with the Prometheus Server you created.

   On the side panel, click **Configuration** > **Data sources** > click the **Add data source** button > select the **Prometheus** data source.

   Make sure to mark the Prometheus data source as **Default**.

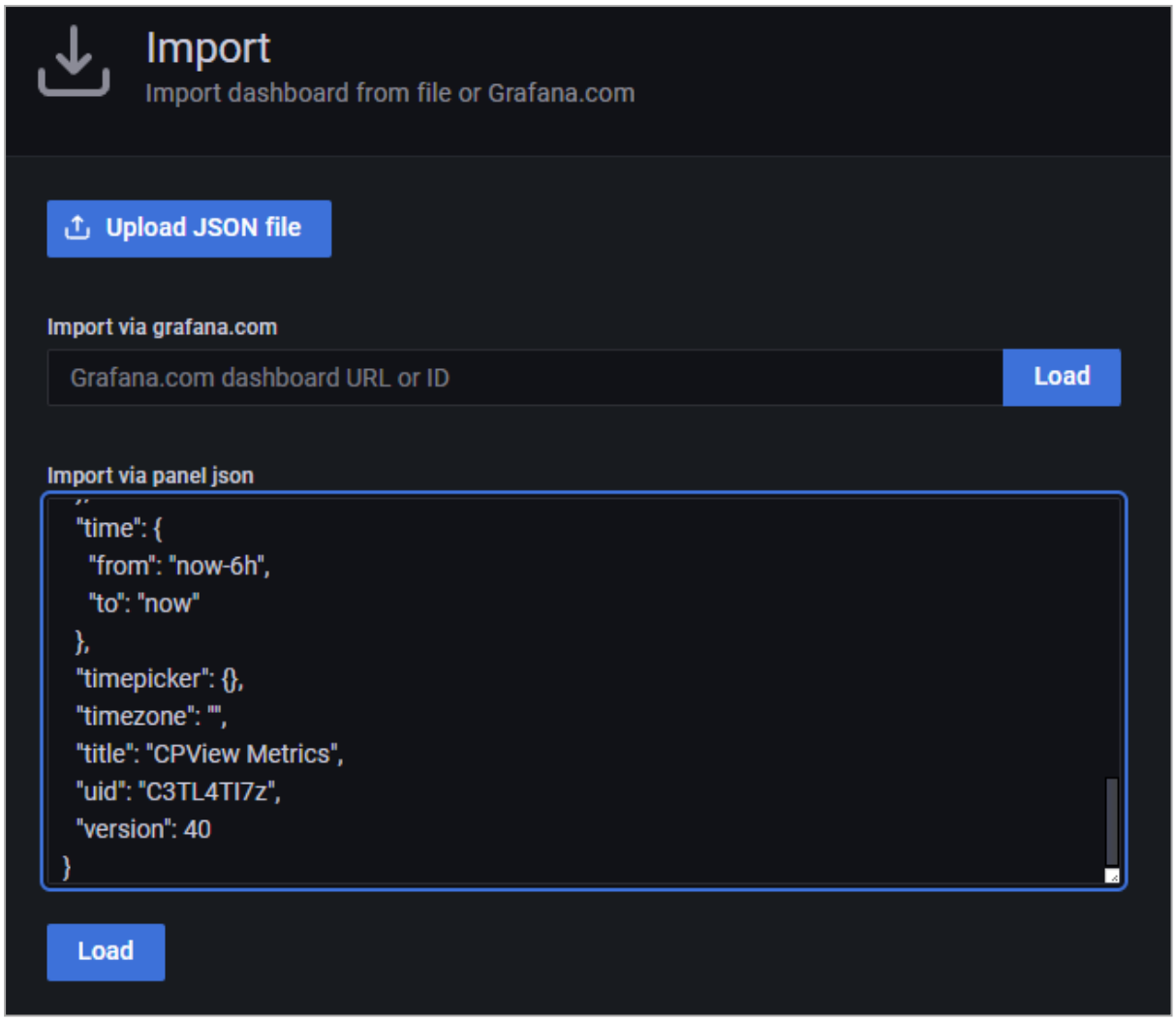
Example:



2. Import the Check Point Grafana dashboard to start monitoring your environment.

   On the side panel, click **Create** > **Import** > upload the applicable JSON file for a dashboard.

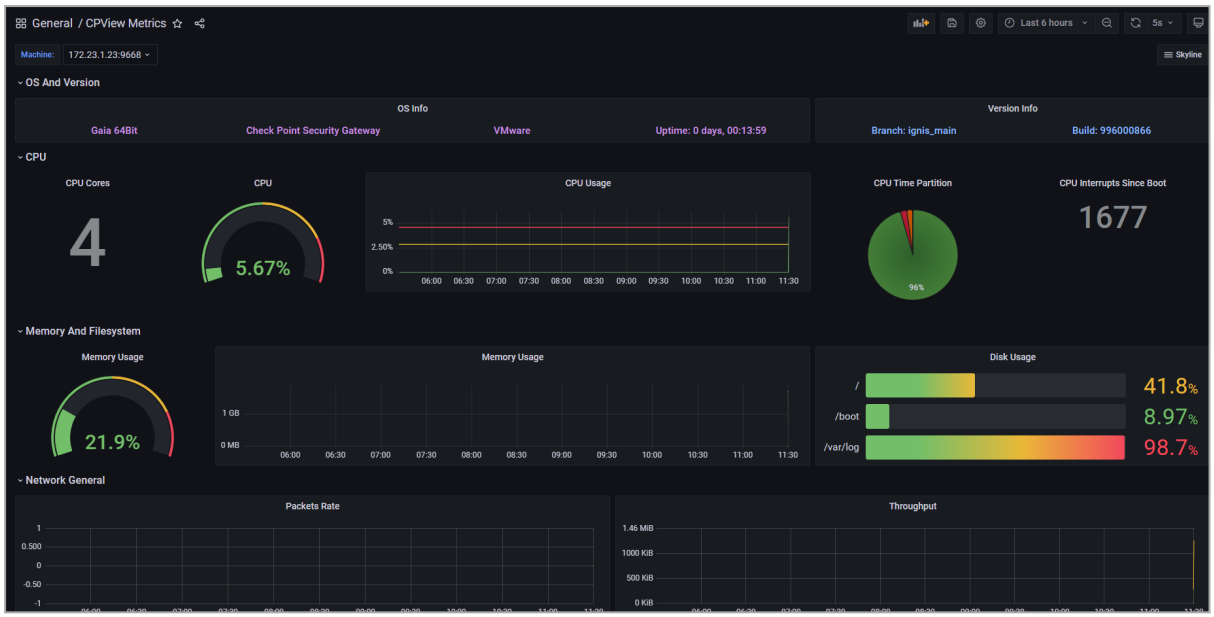
Example:



3. You can modify the existing dashboards or create new dashboards according to your needs.

   To see a full description of all the data exposed by Skyline, see *"Skyline Metrics Repository" on page 64*.

Example:


General / CPView Metrics
Last 6 hours
5s
Machine:
172.23.1.23:9668
Skyline
OS And Version
OS Info
Gaia 64Bit
Check Point Security Gateway
VMware
Uptime: 0 days, 00:13:59
Version Info
Branch: ignis_main
Build: 996000866
CPU
CPU Cores
4
CPU
5.67%
CPU Usage
5%
2.50%
0%
06:00 06:30 07:00 07:30 08:00 08:30 09:00 09:30 10:00 10:30 11:00 11:30
CPU Time Partition
96%
CPU Interrupts Since Boot
1677
Memory And Filesystem
Memory Usage
21.9%
Memory Usage
1 GB
0 MB
06:00 06:30 07:00 07:30 08:00 08:30 09:00 09:30 10:00 10:30 11:00 11:30
Disk Usage
/
41.8%
/boot
8.97%
/var/log
98.7%
Network General
Packets Rate
1
0.500
0
-0.50
-1
Throughput
1.46 MiB
1000 KiB
500 KiB
0 KiB

Step 3 - Install the OpenTelemetry Agent and OpenTelemetry Collector on the Quantum Spark Appliance

Procedure

The Quantum Spark Appliance must be connected to the Internet.

If the Quantum Spark Appliance is connected to the Internet and you enable the Skyline feature, then:

- The appliance automatically downloads the OpenTelemetry Agent (`OtlpAgent`) and the OpenTelemetry Collector (`CPotelcol`) packages.
- The appliance automatically downloads the latest version of the OpenTelemetry Agent package each night.

Step 4 - Configure OpenTelemetry Collector on the Quantum Spark Appliance to work with the Prometheus Server

You can configure the required settings in WebUI or in Gaia Clish.

Procedure in WebUI

1. From the left navigation panel, click **Logs & Monitoring**.
2. In the **Diagnostics** section, click the **OpenTelemetry** page.
3. Select **Monitor this appliance using OpenTelemetry**.
4. In the **URL** field, enter the full path of the receiver endpoint on your monitoring server.

   Example:

   ```
   https://192.168.33.44:9090/api/v1/write
   ```

5. **Optional:** In the **Client Authentication** section, enter the username and password to connect to your monitoring server.
6. **Optional:** In the **Server Authentication** section, upload the CA certificate of your monitoring server:
   a. Click **Upload**.
   b. Click **Browse**.
   c. Select the certificate file and click **Open**.
   d. Click **Preview CA details** to examine the certificate and click **Close**.
   e. Click **Save**.
7. **Optional:** Click **Test Connection** to make sure the correct server and credentials were entered.
8. In the bottom right corner of this page, click **Save**.

Procedure in Gaia Clish

Syntax:

```
set open-telemetry
      cert-base64-encoding "<Base64-Certificate-String>"
      is-active {on | off}
      password <Password>
      url <Full URL>
      username <Username>
```

```
show open-telemetry
```

Syntax Parameters:

Parameter	Description

`cert-base64-encoding`	The Base64-encoded Certificate String for the CA certificate of your monitoring server.
	This is the string that appears in a certificate file between the line "`BEGIN CERTIFICATE`" and the line "`END CERTIFICATE`" (if you take this string from the file, you need to remove the line breaks).
	Example:
	`-----BEGIN CERTIFICATE-----`
	`MIICRzCCAbCgAwIBAg`
	`...(truncated)...`
	`H2IZALydA87zbag=`
	`-----END CERTIFICATE-----`

`is-active`	Enables (`on`) or disables (`off`) the Skyline OpenTelemetry service.
`password`	Specifies the password on your monitoring server.
`url`	Specifies the full path of the receiver endpoint on your monitoring server. Example: `https://192.168.33.44:9090/api/v1/write`
`username`	Specifies the username on your monitoring server.

Instructions:

1. Configure the required settings:

```
set open-telemetry url
"https://192.168.33.44:9090/api/v1/write" username myuser1
password mypassword1 cert-base64-encoding
"MIICRzCCAbCgAwIB...(truncated)...H2IZALydA87zbag=" is-
active on
```

2. Examine the configuration:

```
show open-telemetry
```

Step 5 - Configure Access Policy

You must make sure your Access Policy allows the connection from the Quantum Spark Appliance to the Prometheus Server to send the exported metrics.

On a Locally Managed Quantum Spark Appliance:

You must configure the required settings in WebUI or in Gaia Clish.

See the:

- *R81.10.X Quantum Spark Locally Managed Administration Guide for 1500, 1600, 1800, 1900, 2000 Appliances*.
- *R81.10.X Quantum Spark CLI Reference Guide for 1500, 1600, 1800, 1900, 2000 Appliances*.

On a Centrally Managed Quantum Spark Appliance:

You must configure the required rule on the Management Server (in SmartConsole or with Management API) and install the policy.

See the:

- *R81.10.X Quantum Spark Centrally Managed Administration Guide for 1500, 1600, 1800, 1900, 2000 Appliances*.
- *Quantum Security Management Administration Guide* for your version.
- *Check Point Management API Reference*.

Skyline Metrics Repository

This chapter contains a comprehensive list of all the data available from Skyline - exported from relevant devices using OpenTelemetry protocols.

For each metric, a label representing the context is attached automatically (for example, VS ID in VSX).

To query, filter, and further analyze the trace data, each metric has a <key>:<value> pair known as "Attributes" or "Labels".

This pair represents the metadata of the metric.

Metric data may behave differently with time:

Behavior	Description
Constant	Data is not expected to change in the near future
Slow-Changing	Data might change in very slow intervals (for example, one time during a month)
Fast-Changing	Data is constantly changing
Accumulated	Data is accumulated based on a previous value until a restart

The sections below show:

- Corresponding gauges in the Check Point utility CPView (see sk101878).
- Summary table with the corresponding metric ID and applicable information about it.

References:

- For the metric convention name, refer to https://opentelemetry.io/docs/specs/semconv/general/metrics/.
- For the metric types, refer to https://uptrace.dev/opentelemetry/metrics.html.
- For the unit convention, refer to https://github.com/open-telemetry/semantic-conventions/blob/main/docs/general/metrics.md#instrument-units.

System > CPU

CPView Gauges

1. Run: `cpview`
2. From the top, click **Overview**.

   In the section **CPU**, see:

   - **Load average**.
   - **Num of CPUs**.
3. From the top, click **CPU** > **Overview** > **Host**:
   - In the section **Overview**, see **Avg utilization**.
   - In the section **CPU**, see **Interrupts**.

CLI

Run these commands in the Expert mode:

`top`
`cat /proc/interrupts`

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.cpu.interrupts`	The number of device interrupts that occurred for this CPU core.	Gauge	{interrupts}	{'type', 'cpu'}	■ `type` The category that describes the behavior of the CPU. ■ `cpu` The ID of the CPU core.	The same data for VSX Gateways (VS0) and other Virtual Systems.
`system.cpu.utilization`	The utilization of this CPU core as a percentage of the total utilization.	Gauge	{percent}	{'type', 'cpu', 'state'}	■ `type` The category that describes the behavior of the CPU. ■ `cpu` The ID of the CPU core. ■ `state` Percentage of total time of the CPU code was idle/busy by entity."	The same data for VSX Gateways (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.cpu.count`	The number of CPU cores.	Gauge	{cpu}			The same data for VSX Gateways (VS0) and other Virtual Systems.

System > CPU > Top

CPView Gauges

1. Run: `cpview`
2. From the top, click **CPU** > **Overview** > **Host** > refer to the section **CPU**.
3. From the top, click **CPU** > **Overview** > **Top-Connections** > click the tabs for CoreXL instances > refer to the section **Top Connections**.

CLI

- To see the CPU utilization of the CoreXL Firewall instances, run this command in the Expert mode:

  ```
  top
  ```

- To see the top connections, run this command in the Expert mode (see sk172229):

  ```
  top_conns --help
  ```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`connection.top.cpu.utilization`	CPU utilization of the CoreXL Firewall instance, in %. This metric is available in: ■ R82 and higher	Gauge	{percent}	{'instance', 'protocol', 'connection'}	■ `instance` The number of the CoreXL Firewall instance. ■ `protocol` Protocol of the connection. ■ `connection` IP addresses of the Source and Destination.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`connection.top.cpu.wt.utilization`	"PPE_WT" CPU utilization of the CoreXL SND instance, in %. For information about "PPE_WT", see *Performance Tuning Administration Guide* for your version > Chapter "HyperFlow". This metric is available in: ■ R82 and higher	Gauge	{percent}	{'instance', 'protocol', 'connection'}	■ `instance` The number of the CoreXL SND instance. ■ `protocol` Protocol of the connection. ■ `connection` IP addresses of the Source and Destination.	Different data for each Virtual System.

System > Memory

CPView Gauges

1. Run: `cpview`
2. From the top, click **Overview**.
3. In the section **Memory**, see **Physical**.

CLI

- Run this command in the Expert mode:

  ```
  top
  ```

- Run this command in Gaia Clish:

  ```
  show system memory status
  ```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
system.memory.limit	The total RAM available for processes, in Bytes.	Gauge	By			The same data for VSX Gateway (VS0) and other Virtual Systems.
system.memory.usage	The RAM usage by processes, in Bytes.	AsyncUpDownCounter	By			The same data for VSX Gateway (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.fw.memory.limit`	The total RAM available for Firewall processes, in Bytes. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	Gauge	By			The same data for VSX Gateway (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.fw.memory.usage`	The RAM usage by Firewall processes, in Bytes. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncUpDownCounter	By	{'state'}	■ `state` Memory - "`free`" or "`used`".	The same data for VSX Gateway (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.fw.memory.utilization`	The RAM usage by Firewall processes, in %. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	Gauge	{percent}	{'state'}	■ `state` Used memory in %.	The same data for VSX Gateway (VS0) and other Virtual Systems.

System > Memory Paging

CPView Gauges

1. Run: `cpview`
2. From the top, click **Overview**.
3. In the section **Memory**, see **Swap**.

CLI

- Run this command in the Expert mode:

```
top
```

- Run this command in Gaia Clish:

```
show system memory status
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`system.paging.limit`	The total RAM assigned to swap memory, in Bytes.	Gauge	By		

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`system.paging.usage`	The RAM usage for swap, in Bytes.	AsyncUpDownCounter	By	■ {'state'} Memory - "`free`" or "`used`".	■ `state` The used disk space.

System > Traffic

CPView Gauges

1. Run: `cpview`
2. From the top, click **Network** > **Traffic**:
    a. See the section **Traffic Rate**.
    b. See the section **Concurrent Connections**.
    c. See the section **Drops**.

CLI

Run these commands in Gaia Clish or in the Expert mode (see the *CLI Reference Guide* for your version):

```
cpstat -f policy fw
```

```
fwaccel stats
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.traffic.connections`	Number of concurrent connections	AsyncUpDownCounter	{connections}	{'component', 'state', 'protocol'}	■ `component` The name of the component: • `FW` • `SXL` ■ `state` The current state of the TCP connection: • `handshake` • `established` • `closed` ■ `protocol` The protocol: • `tcp` • `other`	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.traffic.dropped`	The total number of traffic drops made by Security Gateway Software Blades	AsyncCounter	{drops}	{'type'}	▪ `type` The reason for the drop: • `General` • `Rulebase` • `CoreXL` • `SXL` • `Capacity` • `First Packets Not SYN`	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.traffic.dropped.rate`	The rate of traffic drops (number of drops per second) made by Security Gateway Software Blades This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 70 and higher	Gauge	{drops}	{'type', 'ppak_instance', 'component'}	■ `type` The reason for the drop: • `General` • `Rulebase` • `CoreXL` • `SXL` • `Capacity` • `First Packets Not SYN` ■ `ppak_instance` The SecureXL instance associated with the dropped rate. ■ `component` The name of the component responsible for the drop:	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
	■ R81.10 Jumbo Hotfix Accumulator, Take 152 and higher				• FW • SXL	
system.traffic.io.receive	Inbound throughput (bits per second)	Gauge	b/s	{'component'}	■ component The name of the component.	
system.traffic.io.transmit	Outbound throughput (bits per second)	Gauge	b/s	{'component'}	■ component The name of the component.	
system.traffic.packets.receive	Inbound packet rate (packets per second)	Gauge	packets/s	{'component'}	■ component The name of the component.	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.traffic.packets.transmit`	Outbound packet rate (packets per second)	Gauge	packets/sec	{'component'}	▪ `component` The name of the component.	

System > Network

CPView Gauges

1. Run: `cpview`
2. From the top, click **Network** > **Traffic**:
    a. See the section **Traffic Rate**.
    b. See the section **Concurrent Connections**.
    c. See the section **Drops**.

CLI

Run these commands in Gaia Clish or in the Expert mode (see the *CLI Reference Guide* for your version):

```
cpstat -f policy fw
```

```
fwaccel stats
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
system.network.connections	The number of concurrent connections.	AsyncUpDownCounter	{connections}/s	{'component'}	■ component The name of the component.	Different data for each Virtual System.
system.network.connections.rate	Connection rate (connections per second).	Gauge	{connections}/s	{'component'}	■ component The name of the component.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
system.network.tcp_out_of_state_drops.state	The state of the global setting "TCP - Drop Out of State": ■ 0 - disabled ■ 1 - enabled	Gauge	{state}			

System > Network > Interfaces

CPView Gauges

1. Run: `cpview`
2. From the top, click **Network** > **Interfaces**:
   a. Click **Overview**.
   b. Click **Traffic**.

CLI

- To see the configuration of interfaces and their traffic statistics, run these commands in Gaia Clish:

```
show interfaces all
show interface <Name of Interface>
```

- To see the interface driver / firmware, run this command in the Expert mode:

```
show interface <Name of Interface> driver-information
```

- To see the interface driver / firmware, run this command in the Expert mode:

```
ethtool -i <Name of Interface>
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
system.network.interface.state	The state of the network interface: ■ 0 - off ■ 1 - on	Gauge	{state}	{'speed', 'type', 'driver', 'name', 'device', 'port'}	■ speed The current speed of the network interface. ■ type The type of the network interface (for example, "ethernet", "loopback"). ■ driver	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
					The driver name used by this network interface. ■ `name` The name of the network interface as assigned by the operating system. ■ `device`	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
					The name of the network interface device in the operating system. ■ `port` The type of port used by this network interface (for example, "`TP`", "`N/A`").	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.interface.address`	The IP address of the network interface.	Gauge	{address}	{'address', 'protocol'}	■ `address` The name of the network interface device in the operating system. ■ `protocol` The IP version (IPv4 or IPv6).	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.interface.io.receive.rate`	The current rate of successfully received packets over the communication channel (in bits per second).	Gauge	b/s	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.interface.io.receive.rate.peak`	The maximal recorded rate of successfully received packets for this network interface (in bits per second).	Gauge	b/s	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.interface.packets.receive.rate.peak`	The maximal recorded rate of successfully received packets for this network interface (in packets per second)	Gauge	packets/sec	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.interface.packets.receive.rate`	The current rate of successfully received packets over the communication channel (in packets per second).	Gauge	packets/sec	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.interface.io.transmit.rate`	The current rate of successfully transmitted packets over the communication channel (in bits per second).	Gauge	b/s	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.interface.io.transmit.rate.peak`	The maximal recorded rate of successfully transmitted packets for this network interface (in bits per second).	Gauge	b/s	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.interface.packets.transmit.rate.peak`	The maximal recorded rate of successfully transmitted packets for this network interface (in packets per second).	Gauge	packets/sec	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.interface.packets.transmit.rate`	The current rate of successfully transmitted packets over the communication channel (in packets per second).	Gauge	packets/sec	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

System > Network > Packets

CPView Gauges

1. Run: `cpview`
2. From the top, click **Network** > **Interfaces** > **Traffic**.

   See the sections:

   - **RX Traffic**.
   - **TX Traffic**.
   - **Errors and Drops**.
3. From the top, click **Advanced** > **Network** > **Overview**.

   See the section **General Statistics**.

CLI

See the *Gaia Administration Guide* for your version.

- To see the traffic statistics for interfaces, run these commands in Gaia Clish:

```
show interfaces all
show interface <Name of Interface>
```

- To see the interface driver / firmware, run this command in the Expert mode:

```
show interface <Name of Interface> driver-information
```

- To see the interface driver / firmware, run this command in the Expert mode:

```
ethtool -i <Name of Interface>
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.packets.receive`	The total number of received packets by this network interface since the boot.	AsyncCounter	{packets}	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.dropped.receive`	The total number of the received packets that were dropped since boot.	AsyncCounter	{packets}	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.errors.receive`	The total number of corrupted received packets since the boot.	AsyncCounter	{errors}	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.io.receive`	The total number of received traffic bits by the network interface since the boot.	AsyncCounter	b	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.packets.transmit`	The total number of transmitted packets by this network interface since the boot.	AsyncCounter	{packets}	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.io.transmit`	The total number of transmitted traffic bits by the network interface since the boot.	AsyncCounter	b	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.dropped.transmit`	The total number of dropped transmitted packets since the boot.	AsyncCounter	{packets}	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.errors.transmit`	The total number of corrupted transmitted packets since the boot.	AsyncCounter	{errors}	{'device', 'interface'}	■ `device` The name of the network interface device in the operating system. ■ `interface` The name of the network interface as assigned by the operating system.	Different data for each Virtual System.

System > Network > NAT

CPView Gauges

1. Run: `cpview`
2. From the top, click **Advanced** > **NAT**:
    a. Click **Pool-IPv4**.
    b. Click **Pool-IPv6**.

CLI

Run this command in the Expert mode (see the *CLI Reference Guide* for your version):

```
fw ctl pstat -m
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.nat.connections.count`	The number of NAT pool concurrent connections.	AsyncUpDownCounter	{connections}	{'protocol'}	▪ `protocol` Which protocol is used.	Different data for each Virtual System.
`system.network.nat.connections.rate`	The number of NAT pool concurrent connections per second.	AsyncUpDownCounter	{connections}/s	{'protocol'}	▪ `protocol` Which protocol is used.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.nat.ports`	The number of ports used for the NAT pool.	AsyncCounter	{ports}	{'ip_protocol', 'instance', 'type', 'dest_port', 'hide_ip', 'protocol', 'dest_ip'}	■ `ip_protocol` The IP version (IPv4 or IPv6). ■ `instance` The CoreXL Firewall instance, on which the NAT pool is used. ■ `type` The range of the ports. For IPv4: • `low` (600 - 1023)	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
					• high (10000 - 60000) • extra (60001 - 65000) For IPv6: • low6 (600 - 1023) • high6 (10000 - 60000)	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
					• extra6 (60001 - 65000) ■ dest_port The destination port. ■ hide_ip The IP address after NAT. ■ protocol The IP protocol of the connection. ■ dest ip The destination IP address.	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.network.nat.ports.limit`	The total number of ports that can be used for the NAT pool.	Gauge	{ports}	{'ip_protocol', 'instance', 'type', 'dest_port', 'hide_ip', 'protocol', 'dest_ip'}	■ `ip_protocol` The IP version (IPv4 or IPv6). ■ `instance` The CoreXL Firewall instance, on which the NAT pool is used. ■ `type` The range of the ports. For IPv4: • `low` (600 - 1023)	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
					• high (10000 - 60000) • extra (60001 - 65000) For IPv6: • low6 (600 - 1023) • high6 (10000 - 60000)	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
					• extra6 (60001 - 65000) ■ dest_port The destination port. ■ hide_ip The IP address after NAT. ■ protocol The IP protocol of the connection. ■ dest ip The destination IP address.	

System > Filesystem

CPView Gauges

1. Run: `cpview`
2. From the top, click **Overview**.
3. See the section **Disk space (top 3 used partitions)**.

CLI

- Run these commands in Gaia Clish:

```
show system disk usage
show system lvm overview
```

- Run this command in the Expert mode:

```
df -kh
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.filesystem.limit`	Total disk space, in Bytes.	Gauge	By	{'mountpoint'}	■ `mountpoint` The partition mount point.	The same data for VSX Gateways (VS0) and other Virtual Systems.
`system.filesystem.usage`	How much disk space is currently used or free, in Bytes.	AsyncUpDownCounter	By	{'state', 'mountpoint'}	■ `state` Determines whether the value is for the used or the free disk space. ■ `mountpoint` The partition mount point.	The same data for VSX Gateways (VS0) and other Virtual Systems.

System > Input/Output

CPView Gauges

1. Run: `cpview`
2. From the top, click **I/O** > **Overview**.

CLI

Run these commands in the Expert mode:

`top`
`iotop`
`iostat`

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.io.utilization`	Percentage of CPU time during which I/O requests were issued to the device (bandwidth utilization for the device). Device saturation occurs when this value is close to 100%.	Gauge	{percent}	{'device'}	▪ `device` The I/O device name.	The same data for VSX Gateway (VS0) and other Virtual Systems.

System > Gaia

CPView Gauges

1. Run: `cpview`
2. From the top, click **SysInfo**.
3. In the section **Configuration Information**, see:
   - **Platform**.
   - **Configuration**.
4. See the section **Version Information**.

CLI

- Run this command in Gaia Clish:

  ```
  show version all
  ```

- Run this command in the Expert mode on the Multi-Domain Security Management Server (see the *CLI Reference Guide* for your version):

  ```
  $MDSVERUTIL AllCMAs
  ```

Metric Information

For more information, see the *Gaia Administration Guide* for your version.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`system.gaia.module.version`	The branch name and the build number of installed products.	Gauge	{version}	{'module', 'name'}	■ `module` The string "`Firewall-1`". ■ `name` The branch name of the FireWall-1 product.
`system.gaia.os.edition`	The operating system distribution and if the OS kernel is 32-bit or 64-bit.	Gauge	{edition}	{'edition'}	■ `edition` The operating system kernel edition.
`system.gaia.os.role`	The name of the installed Check Point product configuration.	Gauge	{role}	{'role'}	■ `role` The Check Point product configuration (for example, "`Check Point Security Gateway`").
`system.gaia.os.version`	Software release version.	Gauge	{version}	{'version'}	■ `version` The software release version.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`deployment.package.info`	Is this a recommended software version? Note - Check Point marks the relevant software packages as recommended. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	Gauge	{recommended}	{'installation date', 'category', 'file name', 'build'}	■ `installation date` The installation date of the package. ■ `category` The package category: • `major` • `jumbo` • `hotfix` ■ `file name` The file name of the package. ■ `build` The build number of the package.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`env.domain`	Names of the Domains on a Multi-Domain Security Management Server. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	Gauge	{name}	{'name'}	■ `name` The name of the Domain.

System > CoreXL

For more information, see the *Performance Tuning Administration Guide* for your version.

CPView Gauges

1. Run: `cpview`
2. From the top, click **SysInfo**.
3. In the section **Configuration Information**, see:
   - **CoreXL Status**
   - **CoreXL instances**
   - **Dynamic Balancing Status**

CLI

See the *CLI Reference Guide* for your version.

- Run this command in Gaia Clish or in the Expert mode:

```
fw ctl multik stat
```

```
show dynamic-balancing state
```

- Run this command in the Expert mode:

```
dynamic_balancing -p
```

Metric Information

For more information, see the *Performance Tuning Administration Guide* for your version > chapter "CoreXL".

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`firewall.multik.state`	The state of CoreXL: ■ on ■ off	Gauge	{state}			Different data for each Virtual System.
`kernel.instances.count`	The number of CoreXL Firewall instances	Gauge	{instances}			Different data for each Virtual System.
`system.cpu.dynamic_balancing.state`	The state of the CoreXL Dynamic Balancing: ■ On ■ Off	Gauge	{state}			Different data for each Virtual System.

System > SecureXL

For more information, see the *Performance Tuning Administration Guide* for your version.

CPView Gauges

1. Run: `cpview`
2. From the top, click **SysInfo**.

   In the section **Configuration Information**, see: **SecureXL Status**.

   - **SecureXL Status**
   - **UPPAK Status**
3. From the top, click **Advanced** > **SecureXL** > **Backplane** > **Route/ARP**.

   See:

   - **ARP Unresolved**
   - **ipsctl-ppak-route-arp-drops**
4. From the top, click **Software-blades** > **VPN** > **SecureXL**.

   In the section **VPN SecureXL Statistics**, see:

   - **GTP tunnels created**
   - **GTP concurrent tunnels**
   - **GTP accel packets**
   - **GTP F2F packets**
   - **GTP spoofed packets**
   - **GTP signaling packets**

CLI

Run these commands in Gaia Clish or in the Expert mode (see the *CLI Reference Guide* for your version):

```
fwaccel stat
fwaccel stats
```

Metric Information

For more information, see the *Performance Tuning Administration Guide* for your version > chapter "SecureXL".

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`sxl.state`	The state of SecureXL: ■ 0.0 - Off ■ 1.0 - On ■ 2.0 - No license ■ 3.0 - Not initialized ■ -1 - Failed to get the state	Gauge	{state}		

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`adv_prv.expired`	The number of the Hardware Acceleration (adpdrv) ARP Requests that failed to resolve.	Gauge	{requests}		
`adv_prv.errors.count`	Total number of errors related to Hardware Acceleration (adpdrv).	AsyncCounter	{errors}	{'name'}	▪ `name` The name of the error type: • `Service queue is full` • `Nexthop lookup failed` • `ARP resolve failed or queue was full`

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`sxl.gtp.packets`	Total number of GTP packets in SecureXL.	AsyncCounter	{packets}	{'state'}	■ `state` The category of the GTP packets: • `Accelerated` • `F2F` • `Spoofed` • `Signaling`
`sxl.gtp.tunnels.count`	The number of concurrent GTP tunnels in SecureXL.	AsyncUpDownCounter	{tunnels}		
`sxl.gtp.tunnels.created`	Total number of GTP tunnels created in SecureXL.	AsyncCounter	tunnels		

System > SecureXL > SYN Defender

Notes:

- In SecureXL, the name of the feature is "Accelerated SYN Defender".
- In SmartConsole, the name of the applicable IPS protection is "SYN Attack".
- For more information, see the *Performance Tuning Administration Guide* for your version > chapter "SecureXL" > section "Accelerated SYN Defender".

These metrics are available in:

- R82 and higher
- R81.20 Jumbo Hotfix Accumulator Take 41 and higher (PRJ-50104)
- R81.10 Jumbo Hotfix Accumulator Take 131 and higher (PRJ-50103)
- R81 Jumbo Hotfix Accumulator Take 99 and higher (PRJ-50102)
- R80.40 Jumbo Hotfix Accumulator Take 211 and higher (PRJ-50101)

CPView Gauges

1. Run: `cpview`
2. From the top, click **Advanced** > **SecureXL** > **Advanced** > **SYN-Defender**.

CLI

Run this command in Gaia Clish or in the Expert mode (see the *CLI Reference Guide* for your version):

```
fwaccel synatk
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`sxl.synatk.configuration`	SYN Defender Configuration: ■ 0 - Uninitialized ■ 1 - Disabled ■ 2 - Monitoring ■ 3 - Enforcing ■ -1 - Failed to get the state	Gauge	{configuration}		
`sxl.synatk.status`	SYN Defender Status: ■ 0 - Uninitialized ■ 1 - Disabled ■ 2 - Invalid ■ 3 - Under Attack ■ 4 - Attack has just ended ■ 5 - Normal	Gauge	{status}		

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`sxl.synatk.global_high_threshold`	SYN Defender Global High Threshold.	AsyncUpDownCounter	{connections}		
`sxl.synatk.interface_high_threshold`	SYN Defender Interface High Threshold.	AsyncUpDownCounter	{connections}		
`sxl.synatk.low_threshold`	SYN Defender Low Threshold.	AsyncUpDownCounter	{connections}		
`sxl.synatk.ifn_tab.topology`	SYN Defender Interface Topology: ■ 0 - Excluded ■ 1 - Internal ■ 2 - External ■ -1 - Failed to get the topology	Gauge	{topology}	{'name'}	■ `name` The name of the network interface.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
sxl.synatk.ifn_tab.state	SYN Defender state on an interface: ■ 0 - Disabled ■ 1 - Monitor ■ 2 - Ready ■ 3 - Active ■ 4 - Grace ■ -1 - Failed to get the state	Gauge	{state}	{'name'}	■ name The name of the network interface.
sxl.synatk.ifn_tab.duration	SYN Defender Active/Grace Duration.	Gauge	{time}	{'name'}	■ name The name of the network interface.
sxl.synatk.ifn_tab.non_established	Number of SYN Defender Half-Open Connections.	AsyncUpDownCounter	{connections}	{'name'}	■ name The name of the network interface.
sxl.synatk.ifn_tab.sent_cookies	Number of SYN Defender Sent Cookies.	AsyncCounter	{cookies}	{'name'}	■ name The name of the network interface.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`sxl.synatk.ifn_tab.succ_validations`	Number of SYN Defender Successful Validations.	AsyncCounter	{validations}	{'name'}	▪ `name` The name of the network interface.

System > Flofiler

Note - "Flofiler" stands for "Flow Profiler".

CPView Gauges

1. Run: `cpview`
2. From the top, click **Advanced** > **CPU-Profiler**:
    a. Click **Components** > **All-Instances** > **Components**.
    b. Click **PM-Stats** > **All-Instances**.
3. From the top, click **CPU** > **Top-Protocols** > **All-Instances**.

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`flow_profiler.entities`	The number of top CPU consumer entities.	AsycUpDownCounter	{entities}	{'category', 'state'}	■ `category` The Flofiler category: • 0 - Protocol • 1 - Component • 2 - Pattern Matcher ■ `state` There are two states: • `other` The entity is not in the list of top CPU consumers. • `top` The entity is in the list of the top CPU consumers.	Statistics do not exist.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`flow_profiler.utilization`	The CPU utilization for each entity, in %.	Gauge	{percent}	{'category', 'name'}	■ `category` The Flofiler category: • 0 - protocol • 1 - component • 2 - pattern matcher ■ `name` The name of the entity which consumes CPU.	Statistics do not exist.

System > Firewall

CPView Gauges

1. Run: `cpview`
2. From the top, click **SysInfo**.
3. In the section **General information**, see:
    - **Last policy install time**
    - **Last policy name**.

CLI

Run this command in Gaia Clish or in the Expert mode (see the *CLI Reference Guide* for your version):

```
cpstat -f fw policy
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`firewall.policy.name`	The name of the last installed Access Control policy.	Gauge	{name}	{'name'}	▪ `name` The name of the last installed policy.	Statistics exist only in the context of the VSX Gateway (VS0).
`firewall.policy.time`	Time of the last Access Control policy installation.	Gauge	{time}			Statistics exist only in the context of the VSX Gateway (VS0).

System > Process > Top

CPView Gauges

1. Run: `cpview`
2. From the top, click **CPU** > **Processes**.
3. Refer to these sections:
    - **Top 5 Processes - CPU**
    - **Top 5 Processes - RAM**

CLI

Run this command in the Expert mode (and use the applicable keys to sort the required columns):

```
top
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.process.top.cpu.utilization`	CPU utilization by a process, in %. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	Gauge	{percent}	{'pid', 'name', 'state'}	■ `pid` The PID of the process. ■ `name` The name of the process. ■ `state` The state of the process.	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.process.top.fd.count`	Number of file descriptors by a process. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	Gauge	{fd}	{'pid', 'name', 'state'}	■ `pid` The PID of the process. ■ `name` The name of the process. ■ `state` The state of the process.	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`system.process.top.memory.usage`	Memory utilization by a process, in Bytes. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	Gauge	By	{'pid', 'name', 'state'}	■ `pid` The PID of the process. ■ `name` The name of the process. ■ `state` The state of the process.	

System > Process

CPView Gauges

Not available.

CLI

- Run this command in Gaia Clish or in the Expert mode (see the *CLI Reference Guide* for your version.):

```
cpwd_admin list
```

- Run this command in the Expert mode (see https://man7.org/linux/man-pages/man1/ps.1.html):

```
ps
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`cpwd.pid`	PID of the process in Check Point WatchDog. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	{pid}	{'app'}	■ `app` The application in Check Point WatchDog.	The same data for VSX Gateway (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`cpwd.restarts`	Number of times Check Point WatchDog restarted the process. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncCounter	{restarts}	{'app'}	■ `app` The application in Check Point WatchDog.	The same data for VSX Gateway (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`cpwd.state`	State of the process in Check Point WatchDog. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	{state}	{'app'}	■ `app` The application in Check Point WatchDog.	The same data for VSX Gateway (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`cpwd.uptime`	Uptime of the process in Check Point WatchDog, in seconds. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncCounter	s	{'app'}	■ `app` The application in Check Point WatchDog.	The same data for VSX Gateway (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`process.cpu.usage`	CPU utilization by the process, in %. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	{percent}	{'cmd_name', 'process name', 'pid'}	■ `cmd_name` The complete process command line (arguments and the full call list). ■ `process name` The name of the process. ■ `pid` The PID of the process.	The same data for VSX Gateway (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`process.disk.io.read`	Total amount of disk I/O reads by the process, in bits. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncCounter	b	{'cmd_name', 'process name', 'pid'}	■ `cmd_name` The complete process command line (arguments and the full call list). ■ `process name` The name of the process. ■ `pid` The PID of the process.	The same data for VSX Gateway (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`process.disk.io.write`	Total amount of disk I/O writes by the process, in bits. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncCounter	b	{'cmd_name', 'process name', 'pid'}	■ `cmd_name` The complete process command line (arguments and the full call list). ■ `process name` The name of the process. ■ `pid` The PID of the process.	The same data for VSX Gateway (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`process.rss_memory.bytes`	The actual RAM utilization (RSS) by the process, in Bytes. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	By	{'cmd_name', 'process name', 'pid'}	■ `cmd_name` The complete process command line (arguments and the full call list). ■ `process name` The name of the process. ■ `pid` The PID of the process.	The same data for VSX Gateway (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`process.vms_memory.bytes`	The total memory utilization (RAM and Virtual Memory, VMS) by the process, in Bytes. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	By	{'cmd_name', 'process name', 'pid'}	■ `cmd_name` The complete process command line (arguments and the full call list). ■ `process name` The name of the process. ■ `pid` The PID of the process.	The same data for VSX Gateway (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`process.memory.limit`	Memory utilization limit (HWM) by the process, in Bytes. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	By	{'cmd_name', 'process name', 'pid'}	■ `cmd_name` The complete process command line (arguments and the full call list). ■ `process name` The name of the process. ■ `pid` The PID of the process.	The same data for VSX Gateway (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`process.uptime`	Uptime of the process, in seconds. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncCounter	s	{'cmd_name', 'process name', 'pid'}	■ `cmd_name` The complete process command line (arguments and the full call list). ■ `process name` The name of the process. ■ `pid` The PID of the process.	The same data for VSX Gateway (VS0) and other Virtual Systems.

ClusterXL

For more information, see the *ClusterXL Administration Guide* for your version.

CPView Gauges

1. Run: `cpview`
2. From the top, click **Advanced** > **ClusterXL** > **Overview**.

CLI

- Run these commands in Gaia Clish:

```
show cluster state
show cluster members pnotes {all | problem}
```

- Run these commands in the Expert mode:

```
cphaprob state
cphaprob [-l] list
```

Metric Information

For more information, see the *ClusterXL Administration Guide* for your version.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`cluster_xl.members.state`	Current ClusterXL member state.	Gauge	{state}	{'id', 'name'}	■ `id` The ID of the Cluster Member. ■ `name` The name of the Cluster Member. Cluster Member States: ■ 1 - LOST ■ 6 - DOWN ■ 11 - STANDBY ■ 16 - INIT ■ 21 - READY ■ 26 - ACTIVE ■ 27 - ACTIVE (!) ■ 28 - ACTIVE (!F) ■ 29 - ACTIVE (!P) ■ 30 - ACTIVE (!FP) ■ 31 - BACKUP Legend:	Same as ClusterXL

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
					■ ACTIVE (!) - A problem was detected, but the Cluster Member still forwards packets, because it is the only member in the cluster, or because there are no other Active members in the cluster. In any other situation, the state of the member is `Down`. ■ ACTIVE (!F) - See above. The Cluster Member is in the freeze state. ■ ACTIVE (!P) - See above. This is the Pivot Cluster Member in the Load Sharing Unicast mode. ■ ACTIVE (!FP) - See above. This is the Pivot Cluster Member in the Load Sharing Unicast mode and it is in the freeze state.	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`cluster_xl.mode`	The ClusterXL mode: ■ HA - High Availability. ■ LS - Load Sharing ■ Active-Active.	Gauge	{mode}		ClusterXL Modes: ■ 0 - Load Sharing ■ 1 - High Availability Primary Up (the option "Switch to higher priority Cluster Member" in SmartConsole in the ClusterXL object) ■ 2- High Availability Active Up (the option "Maintain current active Cluster Member" in SmartConsole in the ClusterXL object) ■ 3 - Single Virtual System Failover ■ 4 - Virtual System Load Sharing ■ 5 - High Availability over Load Sharing (in Scalable Platforms) ■ 6 - Active-Active ■ 7 - Obsolete (denotes a 3rd-party cluster that only performs Delta Sync without health checks) ■ 8 - Undefined ■ -1 - Failed to get the mode	Same as ClusterXL

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`cluster_xl.pnotes`	Names of Critical Devices that report their state as "problem".	Gauge	{pnotes}	{'pnotes'}	▪ `pnotes` The names of Critical Devices that report their state as "`problem`".	Same as ClusterXL

VSX

For more information, see the *VSX Administration Guide* for your version.

CPView Gauges

1. Run: `cpview`
2. From the top, click **Advanced** > **VSX**:
3. Click **Overview**.
4. Click **VSs** > **Configuration**.
5. Click **VSs** > **Physical-Resources**.
6. Click **VSs** > **Traffic**.
7. Click **Statistics**.

CLI

- Run this command in the Expert mode:

```
vsx stat -v
```

- Run this command in Gaia Clish or in the Expert mode:

```
fw ctl multik stat
```

- Run this command in Gaia Clish:

```
show virtual-system all
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`vsx.core_xl.count`	The number of CoreXL Firewall (FWK) instances configured on the Virtual System.	Gauge	{instances}	{'type', 'name'}	■ `type` Type of the CoreXL Firewall instance: • IPv4 • IPv6 ■ `name` Name of the Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`vsx.overview`	Overview of Virtual Systems - installed policy and SIC Trust.	Gauge	{overview}	{'threat_prevention_policy', 'ac_policy_time', 'type', 'access_control_policy', 'name', 'trust'}	■ `threat_prevention_policy` The name of the Threat Prevention policy installed on the Virtual System. ■ `ac_policy_time` The name of the Access Control policy installed on the Virtual System. ■ `type` Type of the Virtual System: • `MGMT` • `Virtual router` • `sys` • `switch` • `sys_bridge` ■ `access_control_policy`

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
					The time of the Access Control policy installation on the Virtual System. ■ `name` Name of the Virtual System. ■ `trust` The SIC Trust status of the Virtual System.

VoIP

For more information, see the *VoIP Administration Guide* for your version.

CPView Gauges

1. Run: `cpview`
2. From the top, click **Advanced** > **VoIP** > **SIP**.

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`voip.sip.count`	Total number of SIP calls. This metric is available in: ■ R82 and higher	Gauge	{percent}	{'type', 'group'}	■ `type` UDP or TCP. ■ `group` Total or Not inspected.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`voip.sip.earlynat.capacity`	Capacity of the kernel table that holds the ports for SIP Early NAT. This metric is available in: ■ R82 and higher	AsyncUpDownCounter	{percent}			Different data for each Virtual System.
`voip.sip.max.reinv.count`	Maximum number of "re-INVITE" messages per SIP call. This metric is available in: ■ R82 and higher	AsyncUpDownCounter	{messages}	{'type'}	■ `type` REINV.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`voip.sip.multicore.state`	Whether the VoIP SIP MultiCore feature is enabled or disabled (see sk180394): ■ 0.0 - SIP MultiCore is disabled ■ 1.0 - SIP MultiCore is enabled (default) This metric is available in: ■ R82 and higher	Gauge	{state}			Different data for each Virtual System.

Network > Heavy Connections

CPView Gauges

1. Run: `cpview`
2. From the top, click **CPU** > **Top-Connections** > **InstancesX-Y** > **InstanceZ**.
3. Refer to the section **Top Connections**.

CLI

See the *CLI Reference Guide* for your version > Chapter "CoreXL Commands" > Section "fw ctl multik" > Section "fw ctl multik print_heavy_conn".

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`network.heavy_connection.packets`	Number of packets that were transferred in the connection. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	Gauge	{packets}	{'start_time', 'conn', 'service', 'identification_time', 'duration', 'instance_load', 'connection_instance_load'}	■ `start_time` The time when the heavy connection started. ■ `conn` The connection tuple - Source IP, Destination IP, Destination Port, and Protocol. ■ `service` Protocol and Port. ■ `identification_time` The time when the connection was recognized as heavy. ■ `duration` For how long the connection lasted, in seconds. ■ `instance_load` The CPU utilization of the CoreXL Firewall instance.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
					■ `connection_instance_load` The part of the connection in the CPU utilization of the CoreXL Firewall instance.	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`network.heavy_connection.bytes`	Number of bytes that were transferred in the connection. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	Gauge	{bytes}	{'start_time', 'conn', 'service', 'identification_time', 'duration', 'instance_load', 'connection_instance_load'}	■ `start_time` The time when the heavy connection started. ■ `conn` The connection tuple - Source IP, Destination IP, Destination Port, and Protocol. ■ `service` Protocol and Port. ■ `identification_time` The time when the connection was recognized as heavy. ■ `duration` For how long the connection lasted, in seconds. ■ `instance_load` The CPU utilization of the CoreXL Firewall instance.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
					■ `connection_instance_load` The part of the connection in the CPU utilization of the CoreXL Firewall instance.	

Network > Network Probes (VPN)

On Security Gateways R82 and higher, you can configure Network Probes to monitor the status of Site-to-Site VPN Tunnels. See sk181994.

For more information, see the *Site to Site VPN Administration Guide* for your version.

In addition, see:

- *"VPN > Probes" on page 182*.
- *"Blades > VPN" on page 194*.

CLI

Run this command in the Expert mode:

```
probemon help
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`network.probes.state`	The state of the Network Probe. This metric is available in: ■ R82 and higher	Gauge	{state}	{'id', 'name', 'destination', 'source'}	■ `id` The ID of the Network Probe. ■ `name` The name of the Network Probe. ■ `destination` The destination address of the Network Probe. ■ `source` The source address of the Network Probe.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`network.probes.mode`	The monitoring mode of the Network Probe: ■ ICMP ■ HTTP This metric is available in: ■ R82 and higher	Gauge	{mode}	{'id'}	■ `id` The ID of the Network Probe.	Different data for each Virtual System.
`network.probes.last_probe`	The time of the last report from the Network Probe. This metric is available in: ■ R82 and higher	Gauge	{second}/s	{'id'}	■ `id` The ID of the Network Probe.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`network.probes.last_status_change`	The time of the last change in the state of the Network Probe. This metric is available in: ■ R82 and higher	Gauge	{second}/s	{'id'}	■ `id` The ID of the Network Probe.	Different data for each Virtual System.

VPN > Probes

On Security Gateways R82 and higher, you can configure Network Probes to monitor the status of Site-to-Site VPN Tunnels. See sk181994.

For more information, see the *Site to Site VPN Administration Guide* for your version.

In addition, see:

- *"Network > Network Probes (VPN)" on page 178*.
- *"Blades > VPN" on page 194*.

CLI

Run these commands in the Expert mode (see the *CLI Reference Guide* for your version):

```
probemon help
vpn tu tlist
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`vpn.probes.tunnel`	The state of the VPN tunnel: ■ 0 - Down ■ Any positive integer - Up (this value is the outbound SPI of the VPN tunnel) This metric is available in: ■ R82 and higher	Gauge	{tunnel}	{'id', 'mspi', 'peer_name', 'peer_main_ip', 'remote_ip', 'local_ip', 'tunnel_methods', 'local_ts', 'remote_ts', 'tunnel_type'}	■ `id` The ID of the Network Probe. ■ `mspi` The MSPI of the VPN tunnel. ■ `peer_name` The name of the VPN peer. ■ `peer_main_ip` The main IP address of VPN peer. ■ `remote_ip` The IP address of VPN peer for this VPN tunnel. ■ `local_ip`	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
					The local IP address for this VPN tunnel. ■ `tunnel_methods` VPN tunnel encryption methods. ■ `local_ts` Local Traffic Selector (destination subnets). ■ `remote_ts` Traffic Selector on the VPN peer (destination subnets). ■ `tunnel_type` The VPN tunnel type (SSL, NAT-T, Visitor Mode, and so on).	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`vpn.probes.tunnel_generated_time`	The time when the VPN Tunnel was established. This metric is available in: ■ R82 and higher	Gauge	{second}/s	{'id'}	■ `id` The ID of the Network Probe.	Different data for each Virtual System.
`vpn.probes.tunnel_expire_time`	The time when the VPN Tunnel will expire. This metric is available in: ■ R82 and higher	Gauge	{second}/s	{'id'}	■ `id` The ID of the Network Probe.	Different data for each Virtual System.
`vpn.probes.tunnel_encrypted_bytes`	The total number of encrypted bytes. This metric is available in: ■ R82 and higher	Gauge	By	{'id'}	■ `id` The ID of the Network Probe.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`vpn.probes.tunnel_encrypt_throughput`	VPN tunnel encrypted throughput. This metric is available in: ■ R82 and higher	Gauge	b/s	{'id'}	■ `id` The ID of the Network Probe.	Different data for each Virtual System.
`vpn.probes.tunnel_decrypted_bytes`	The total number of decrypted bytes. This metric is available in: ■ R82 and higher	Gauge	By	{'id'}	■ `id` The ID of the Network Probe.	Different data for each Virtual System.
`vpn.probes.tunnel_decrypt_throughput`	VPN tunnel decrypted throughput. This metric is available in: ■ R82 and higher	Gauge	b/s	{'id'}	■ `id` The ID of the Network Probe.	Different data for each Virtual System.

Blades > Status and Update

CPView Gauges

1. Run: `cpview`
2. From the top, click **Software-blades** > **Overview**.

CLI

Run these commands in Gaia Clish or in the Expert mode (see the *CLI Reference Guide* for your version):

`cpstat`
`cpstat -f <Flavor> <Feature>`

Software Blade	Command to See the Status

Application Control	■ To the Software Blade status:
	`cpstat -f default appi`
	■ To the Software Blade subscription status:
	`cpstat -f subscription_status appi`
	■ To the Software Blade update status:
	`cpstat -f update_status appi`

Software Blade	Command to See the Status

URL Filtering	■ To the Software Blade status:
	`cpstat -f default urlf`
	■ To the Software Blade subscription status:
	`cpstat -f subscription_status urlf`
	■ To the Software Blade update status:
	`cpstat -f update_status urlf`

Anti-Virus
and
Anti-Bot
	■ To the Software Blade status:
	`cpstat -f default antimalware`
	■ To the Software Blade subscription status:
	`cpstat -f subscription_status antimalware`
	■ To the Software Blade update status:
	`cpstat -f update_status antimalware`

Threat Extraction	■ To the Software Blade status and engine version:
	`cpstat -f default scrub`
	■ To the Software Blade subscription status:
	`cpstat -f subscription_status scrub`

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`blades.state`	Is the Software Blade active? ■ 0 - Enabled ■ 1 - Disabled ■ 2 - Unknown This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	{state}	{'name'}	■ `name` The name of the Software Blade.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`blades.entitlement`	Is the Software Blade entitled to download updates? ■ 0 - Entitled ■ 1 - Not entitled ■ 2 - Evaluation ■ 3 - Expired ■ 4 - Unavailable ■ 5 - Unknown status ■ 6 - Not applicable This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	{entitlement}	{'name'}	■ `name` The name of the Software Blade.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`blades.expiration`	When does the Software Blade entitlement expire (in seconds)? This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	s	{'name'}	■ `name` The name of the Software Blade.
`blades.update.state`	Update status of Software Blades.	Gauge	{state}	{'entity'}	■ `entity` The name of the Software Blade.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`blades.update.time`	Update time of Software Blades.	Gauge	{time}	{'entity', 'name'}	■ `entity` The name of the Software Blade. ■ `name` The time of the last update.

Blades > VPN

For more information, see the *Site to Site VPN Administration Guide* for your version.

In addition, see:

- *"Network > Network Probes (VPN)" on page 178*.
- *"VPN > Probes" on page 182*.

CPView Gauges

1. Run: `cpview`
2. From the top, click **Software-blade** > **VPN**.
3. Click **Overview**.
4. Click **Detailed**.
5. Click **Tunnel-Monitoring**.
6. Click **SecureXL**.

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
system.network.blades.vpn.active_clients	Current number of connected Remote Access VPN clients.	AsyncUpDownCounter	{Counter}			The same data for VSX Gateways (VS0) and other Virtual Systems.
system.network.blades.vpn.all_ike_errors	Total number of all IKE errors.	AsyncCounter	{errors}			The same data for VSX Gateways (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
system.network.blades.vpn.ike_sas	Number of concurrent IKE SAs.	AsyncUpDownCounter	{sas}			The same data for VSX Gateways (VS0) and other Virtual Systems.
system.network.blades.vpn.kernel_limit_reached_count	Sum of values from the VPN limit counters (ike2peer_reach_limit) and (vpn_queues_reach_limit).	AsyncUpDownCounter	{errors}			The same data for VSX Gateways (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
system.network.blades.vpn.max_ike_sas	Maximum number of IKE SAs that this Security Gateway initiated.	Gauge	{sas}			The same data for VSX Gateways (VS0) and other Virtual Systems.
system.network.blades.vpn.total_sas	Total number of IKE SAs.	AsyncCounter	{sas}			The same data for VSX Gateways (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`vpn.clients`	Current number of connected Remote Access VPN clients.	AsyncUpDownCounter	{clients}	{'mode'}	▪ `mode` The mode associated with the client: • `Office` • `Visitor` • `SNX` • `L2TP`	The same data for VSX Gateways (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`vpn.compression.bytes`	VPN Data related to packets compression.	AsyncCounter	By	{'state'}	▪ `state` The compression status: • `Before Compression` • `After Compression` • `Compression Overhead` • `Non Compressed`	The same data for VSX Gateways (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`vpn.compression.packets`	VPN Data related to packets compression.	AsyncCounter	{packets}	{'state'}	▪ `state` The compression status: • `Before Compression` • `After Compression` • `Compression Overhead` • `Non Compressed`	The same data for VSX Gateways (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`vpn.errors`	VPN errors (includes ESP, ESP/UDP, fragmentation, and other errors).	AsyncCounter	{errors}	{'type', 'state'}	■ `type` The type of the error: • `General` • `IPSEC` • `IKE` • `Accelerated VPN` ■ `state` The reason for this error. For example: • `Encrypted` • `Decrypted` • `Should have been encrypted` • `Should be clear text packets` • `Authentication`	The same data for VSX Gateways (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
					• No response from peer	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
vpn.ike.concurrent	Concurrent IKE SAs.	AsyncUpDownCounter	{sas}	{'type'}	■ type The IKE SA type: • IKE v1 • IKE v2 • IKE SAs Interoperable Devices • IKE SAs Dynamically Assigned IP addresses • IKE SAs Remote Access Endpoints • IKE SAs IPv6 • IKE SAs By Machine • IKE SAs By Peer	The same data for VSX Gateways (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
					• Concurrent IKE SA negotiations • Concurrent IKE SA exchange attempts • Concurrent IKE SA exchange attempts by machine • Concurrent IKE SAs with aggressive mode	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`vpn.ike.count`	Total number of IKE SAs.	AsyncCounter	{sas}	{'type'}	▪ `type` The SA type: • `SAs Init By Machine` • `SAs Init By Peer` • `IPSec Inbound SAs` • `IPSec Outbound SAs`	The same data for VSX Gateways (VS0) and other Virtual Systems.
`vpn.ike.max`	Maximum number of concurrent IKE SAs that this Security Gateway initiated.	Gauge	{sas}	{'type'}	▪ `type` The SA type: • `SAs Init By Machine` • `SAs Init By Peer`	The same data for VSX Gateways (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
vpn.ike.negotiations.max	Maximum number of concurrent IKE SA negotiations.	Gauge	{sas}			The same data for VSX Gateways (VS0) and other Virtual Systems.
vpn.ike.peers	Number of concurrent IKE peers.	Gauge	{peers}			The same data for VSX Gateways (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
vpn.ioctls	Total number of kernel IOCTL calls.	AsyncCounter	{ioctls}			The same data for VSX Gateways (VS0) and other Virtual Systems.
vpn.ipsec.fragmentation.count	Number of fragmentation's caused due to IPsec.	AsyncCounter	{fragmentations}			The same data for VSX Gateways (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
vpn.ipsec.fragmentation.drops	Total number of times the Security Gateway dropped traffic that was fragmented because of IPsec.	AsyncCounter	{drops}			The same data for VSX Gateways (VS0) and other Virtual Systems.
vpn.kernel_traps	Total number of kernel trap calls.	AsyncCounter	{traps}			The same data for VSX Gateways (VS0) and other Virtual Systems.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`vpn.packets`	Total number of VPN related packets since the last boot.	AsyncCounter	{packets}	{'type'}	■ `type` The type of packets: • `Encrypted` • `Decrypted`	The same data for VSX Gateways (VS0) and other Virtual Systems.
`vpn.restarts`	Total number of VPN restarts or VPN policy reloads.	AsyncCounter	{restarts}	{'type', 'name'}	■ `type` What kind of entity is counted: • `Process` • `Policy` ■ `name` The name of the entity: • `VPND` • `IKED`	The same data for VSX Gateways (VS0) and other Virtual Systems.

Blades > IDA

For more information, see the *Identity Awareness Administration Guide* for your version.

CPView Gauges

1. Run: `cpview`
2. From the top, click **Software-blades** > **IDA**.

CLI

Run these commands in the Expert mode (see the *CLI Reference Guide* for your version):

`adlog`
`pdp`
`pep show`

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`ida.ad_query.events`	AD Query data.	AsyncCounter	{events}	{'ip', 'name'}	▪ `ip` The destination IP address of the AD query. ▪ `name` The name of the AD query.	Different data for each Virtual System.
`ida.ad_query.state`	AD Query state.	Gauge	{state}	{'ip', 'name'}	▪ `ip` The destination IP address of the AD query. ▪ `name` The name of the AD query.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`ida.authenticated`	Total number of authenticated objects that Identity Awareness acquired.	AsyncCounter	{objects}	{'entity', 'method'}	▪ `entity` The owner (user or machine). ▪ `method` The authentication method: • Kerberos • User and password • AD Query • SAML	Different data for each Virtual System.
`ida.authenticated.count`	Total number of authenticated objects that Identity Awareness acquired.	AsyncCounter	{objects}	{'entity'}	▪ `entity` The owner (user or machine).	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`ida.components.state`	The state of the Identity Awareness component (PDP, or PEP).	Gauge	{state}	{'local', 'name'}	▪ `local` Is the component running on the Security Gateway you query? ▪ `name` The name of the PDP or PEP server.	Different data for each Virtual System.
`ida.components.disconnections`	Number of times the component disconnected.	AsyncCounter	{disconnections}	{'name'}	▪ `name` The name of the PDP or PEP server.	Different data for each Virtual System.
`ida.memory`	Memory used by Identity Awareness.	AyncCounter	By	{'type', 'entity'}	▪ `type` The type of the buffer. ▪ `entity` The kind of session.	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`ida.logins.successful`	Total number of successful logins related to Identity Awareness.	AsyncCounter	{logins}	{'entity', 'method'}	▪ `entity` The login owner (user or machine). ▪ `method` The authentication method: • Kerberos • User and password	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`ida.logins.count`	Total number of logins related to Identity Awareness.	AsyncCounter	{logins}	{'method'}	▪ `method` The authentication method: • Captive Portal • Total Identity Agents • AD Query • Full Identity Agents for Windows OS • Lite Identity Agents for Windows OS • Mac Identity Agents for macOS	Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
					• Identity Agents for Windows Terminal Server • RADIUS Accounting • Remote Access VPN • Identity Collectors (Active Directory) • Identity Collectors (Cisco ISE) • Identity Collectors (eDirectory) • Identity Awareness API	

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
					• Identity Awareness acquired from the Packet Tagging protection	
ida.unauthenticated	Total number of unauthenticated objects that Identity Awareness acquired.	AsyncCounter	{objects}			Different data for each Virtual System.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`ida.logged.unsuccessful`	Total number of unsuccessful logins related to Identity Awareness.	AsyncCounter	{logins}	{'entity', 'method'}	▪ `entity` The owner (user or machine). ▪ `method` The authentication method: • Kerberos • User and password	Different data for each Virtual System.
`ida.user_directory.count`	Total number of User Directory queries.	AsyncCounter	{queries}	{'state'}	▪ `state` Did the query succeed?	Different data for each Virtual System.

Hardware > Model

For information about hardware thresholds on Check Point Appliances, see sk119232.

For hardware documentation for your Check Point Appliance, see sk96246.

CPView Gauges

1. Run: `cpview`
2. From the top, click **SysInfo**.
3. In the section **Hardware Information**, see **HW Model**.

CLI

Run the "`dmidecode`" command in the Expert mode as described in sk37692.

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`hardware.model`	The model name of the Check Point Appliance or Open Server.	Gauge	{model}	{'model'}	▪ `model` The model name of the Check Point appliance or Open Server.	Statistics exist only in the context of the VS0.

Hardware > Temperature

For information about hardware thresholds on Check Point Appliances, see sk119232.

CPView Gauges

1. Run: `cpview`
2. From the top, click **Hardware-Health** > **Temperature**.

CLI

Run this command in Gaia Clish:

```
show sysenv temp
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`hardware.temperature.max`	The maximal temperature that is valid	Gauge	Cel	{'name'}	▪ `name` The name of the temperature sensor.	Statistics exist only in the context of the VS0.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`hardware.temperature.min`	The minimal temperature that is valid.	Gauge	Cel	{'name'}	▪ `name` The name of the temperature sensor.	Statistics exist only in the context of the VS0.
`hardware.temperature.state`	The state of the sensor: ▪ 0 - Works correctly ▪ 1 - Failed	Gauge	{state}	{'name'}	▪ `name` The name of the temperature sensor.	Statistics exist only in the context of the VS0.
`hardware.temperature`	The current temperature measurement of the sensor.	AsyncUpDownCounter	Cel	{'name'}	▪ `name` The name of the temperature sensor.	Statistics exist only in the context of the VS0.

Hardware > Voltage

For information about hardware thresholds on Check Point Appliances, see sk119232.

CPView Gauges

1. Run: `cpview`
2. From the top, click **Hardware-Health** > **Voltage**.

CLI

Run this command in Gaia Clish:

```
show sysenv volt
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`hardware.voltage.min`	The minimal voltage value that is valid.	Gauge	V	{'name'}	▪ `name` The name of the voltage sensor.	Statistics exist only in the context of the VS0.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`hardware.voltage.max`	The maximal voltage value that is valid.	Gauge	V	{'name'}	■ `name` The name of the voltage sensor.	Statistics exist only in the context of the VS0.
`hardware.voltage.state`	The state of the sensor: ■ 0 - Works correctly ■ 1 - Failed	Gauge	{state}	{'name'}	■ `name` The name of the voltage sensor.	Statistics exist only in the context of the VS0.
`hardware.voltage`	The current voltage measurement of the sensor.	AsyncUpDownCounter	V	{'name'}	■ `name` The name of the voltage sensor.	Statistics exist only in the context of the VS0.

Hardware > PSU

For information about hardware thresholds on Check Point Appliances, see sk119232.

For hardware documentation for your Check Point Appliance, see sk96246.

CPView Gauges

1. Run: `cpview`
2. From the top, click **Hardware-Health** > **Power-Suppy**.

CLI

Run this command in Gaia Clish:

```
show sysenv ps
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`hardware.power_supply`	State of the PSU: ■ 0.0 - Down ■ 1.0 - Up ■ 2.0 - Empty ■ 3.0 - Dummy This metric is available in: ■ R82 and higher	Gauge	{state}	{'name'}	■ `name` The state value.	Statistics exist only in the context of the VS0.
`hardware.power_supply.state`	Which PSU is used: ■ 0.0 - Secondary ■ 1.0 - Primary (default)	Gauge	{state}	{'name'}	■ `name` Primary or Secondary.	Statistics exist only in the context of the VS0.

Hardware > Fans

For information about hardware thresholds on Check Point Appliances, see sk119232.

For hardware documentation for your Check Point Appliance, see sk96246.

CPView Gauges

1. Run: `cpview`
2. From the top, click **Hardware-Health** > **Fan**.

CLI

Run this command in Gaia Clish:

```
show sysenv fans
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`hardware.fan`	The current fan speed.	AsyncUpDownCounter	RPM	{'name'}	▪ name The name of the Fan Unit.	Statistics exist only in the context of the VS0.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
hardware.fan.max	The maximal value that is valid.	Gauge	RPM	{'name'}	▪ name The name of the Fan Unit.	Statistics exist only in the context of the VS0.
hardware.fan.min	The minimal value that is valid.	Gauge	RPM	{'name'}	▪ name The name of the Fan Unit.	Statistics exist only in the context of the VS0.
hardware.fan.state	The state of the sensor: ▪ 0 - Works correctly ▪ 1 - Failed	Gauge	{state}	{'name'}	▪ name The name of the Fan Unit.	Statistics exist only in the context of the VS0.

Hardware > BIOS

For information about hardware thresholds on Check Point Appliances, see sk119232.

1. Run: `cpview`
2. From the top, click **Hardware-Health** > **Bios**.

CLI

Run this command in Gaia Clish:

```
show sysenv bios
```

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`hardware.bios`	Which BIOS is used: ■ 0.0 - Secondary ■ 1.0 - Primary (default) This metric is available in: ■ R82 and higher	Gauge	{state}	{'name'}	■ `name` Primary or Secondary.	Statistics exist only in the context of the VS0.
`hardware.bios.state`	State of BIOS: ■ 1.0 - Up ■ 0.0 - Down	Gauge	{state}	{'name'}	■ `name` Up or Down.	Statistics exist only in the context of the VS0.

OtlpAgent > Scripts

For more information about this feature, see *"Custom Metrics" on page 265*.

For more information about OpenTelemetry Agent (OtlpAgent), see sk181615.

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`script.cpu.usage`	CPU utilization by the custom script, in %. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	{percent}	{'command', 'interval', 'output size'}	■ `command` The name of the custom script file. ■ `interval` The interval to run the custom script. ■ `output size` The size of the custom script's output.	Depends on the custom script.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`script.memory.usage`	Memory utilization by the custom script, in Megabytes. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	{Mb}	{'command', 'interval', 'output size'}	■ `command` The name of the custom script file. ■ `interval` The interval to run the custom script. ■ `output size` The size of the custom script's output.	Depends on the custom script.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`script.runtime`	Runtime of the custom script, in seconds. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	s	{'command', 'interval', 'output size'}	■ `command` The name of the custom script file. ■ `interval` The interval to run the custom script. ■ `output size` The size of the custom script's output.	Depends on the custom script.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`script.status`	Status of the custom script: ■ 0.0 - Succeeded ■ 1.0 - Failed ■ 2.0 - Terminated ■ 3.0 - CPU utilization exceeded the threshold of 5% (you cannot change this threshold) ■ 4.0 - Memory utilization exceeded the threshold of 10MB (you cannot change this threshold)	AsyncGauge	{status}	{'command', 'interval', 'output size'}	■ `command` The name of the custom script file. ■ `interval` The interval to run the custom script. ■ `output size` The size of the custom script's output.	Depends on the custom script.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
	■ 5.0 - Runtime exceeded the threshold of 5 minutes (you cannot change this threshold) ■ -1 - Error with the custom metric configuration					

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
	This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher					

OtlpAgent > Version

For more information about OpenTelemetry Agent (OtlpAgent), see sk181615.

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description	VSX Behavior
`otlp_agent.version`	The version of the Skyline OpenTelemetry Agent (OtlpAgent). This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	{version}	{'version', 'branch'}	■ `version` The version of the agent. ■ `branch` The software branch of the agent.	The same data for VSX Gateway (VS0) and other Virtual Systems.

API on the Management Server

For more information, see the *Check Point Management API Reference*.

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`mgmt.proc.ready`	Are the require processes ready on the Management Server? This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	{is_ready}	{'process_ name', 'state, 'more'}	■ `process_name` The name of the process (FWM, CPM, API, and so on). ■ `state` The state of the process (Up or Down). ■ `more` An additional description for the process status.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`api.call.count`	Number of calls to each API command. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	{count}	{'command'}	■ `command` The name of the API command.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`api.call.success.rate`	Success rate of each API command. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	{rate}	{'command'}	■ `command` The name of the API command.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`api.call.avg.duration`	Average duration of each API command. This metric is available in: ■ R82 and higher ■ R81.20 Jumbo Hotfix Accumulator, Take 54 and higher ■ R81.10 Jumbo Hotfix Accumulator, Take 135 and higher ■ R81 Jumbo Hotfix Accumulator, Take 99 and higher	AsyncGauge	{duration}	{'command'}	■ `command` The name of the API command.

Maestro Orchestrator

CPView Gauges

1. Run: `cpview`
2. From the top, click **Orchestrator** > click the applicable tab.

CLI

On the Orchestrator, run these commands in the Expert mode:

`orch_stat -h`
`orch_stat --LLDP`
`orch_stat --lag [--verbose]`
`orch_stat --ports`

Metric Information

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`orchestrator.deployment.member_id`	The Member ID of this Orchestrator on its Maestro Site. This metric is available in: ■ R82 and higher	Gauge		{'device_type', 'serial', 'mac_address', 'site_id'}	■ `device_type` The hardware model of the Orchestrator. ■ `serial` The serial number of the Orchestrator. ■ `mac_address` The MAC address of the Orchestrator. ■ `site_id` The ID of the Maestro Site.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`orchestrator.deployment.num_of_mhos_on_site`	The number of Orchestrators on this Maestro Site. This metric is available in: ■ R82 and higher	Gauge		{'site_id'}	■ `site_id` The ID of the Maestro Site.
`orchestrator.deployment.num_of_sites`	The number of Maestro Sites. This metric is available in: ■ R82 and higher	Gauge		{'site_id'}	■ `site_id` The ID of the Maestro Site.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`orchestrator.deployment.orchd`	The status of the main daemon `orchd` on this Orchestrator: ■ 0.0 - Down ■ 1.0 - Up (default) This metric is available in: ■ R82 and higher	Gauge			
`orchestrator.deployment.site_id`	The ID of the Maestro Site. This metric is available in: ■ R82 and higher	Gauge		{'site_id'}	■ `site_id` The ID of the Maestro Site.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`orchestrator.lldp`	Information about the LLDP messages received from Security Appliances. This metric is available in: ■ R82 and higher	AsyncCounter	{state}	{'physical_port', 'logical_port', 'hostname', 'type', 'id', 'ip', 'port_id', 'hw_type', 'state', 'version', 'remote_interface'}	■ `physical_port` The ID of the physical port on the Orchestrator (in the format "<Orchestrator ID>/<Port Label>/<Subport Label>"). ■ `logical_port` The ID of the logical port on the Orchestrator (number "<Port Label>" from the physical port ID). ■ `hostname` The hostname of the Security Appliance. ■ `type`

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
					The type (role) of port on the Orchestrator ("`GW`" or "`Orchestrator`"). ■ `id` Either the ID of the Security Appliance, or the Member ID of the Orchestrator on this Maestro Site. ■ `ip` The IP Address of the Security Appliance. ■ `port_id` The internal port ID. ■ `hw_type` The hardware model of the Security Appliance.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
					■ `state` The state of the Security Appliance ("`ACTIVE`" or "`UNASSIGNED`"). ■ `version` The software version on the Security Appliance. ■ `remote_interface` The name of the interface on the Security Appliance.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`orchestrator.lag`	The current states of Bond (LAG) interfaces for communication with peer Orchestrators - on the same site and on the peer site. This metric is available in: ■ R82 and higher	AsyncUpDownCounter	{state}	{'status', 'lag_id', 'lag_id_full', 'lag_name', 'sync_type', 'hostname', 'link_state', 'logical_port', 'membership_state', 'physical_port', 'port_id'}	■ `status` The status of the Bond interface ("`UP`" or "`DOWN`"). ■ `lag_id` The LAG ID. ■ `lag_id_full` The LAG full ID. ■ `lag_name` The LAG name ("`SYNC-INT`" or "`SYNC-EXT`"). ■ `sync_type` The sync type - "`Local Site`" (internal sync on the same Site) or "`Remote Site`" (external sync between Sites). ■ `hostname`

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
					The hostname of the peer Orchestrator. ■ link_state The status of the Bond interface link ("UP" or "DOWN"). ■ logical_por The logical port ID ("<Port ID>" or "<Split ID>") ■ membership_ state The LAG Membership State ("Active" or "Inactive"). ■ physical_ port

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
					The physical port ID (in the format "<Orchestrator ID>/<Port ID>/<Split ID>"). ■ `port_id` The internal port ID.
`orchestrator.ports.admin_state`	The administrative state of the port on this Orchestrator: ■ 0.0 - Down ■ 1.0 - Up (default) This metric is available in: ■ R82 and higher	Gauge	{state}	{'name', 'logical_port'}	■ `name` The name of the port on the Orchestrator. ■ `logical_port` The ID of the logical port on the Orchestrator (number "<Port Label>" from the physical port ID).

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`orchestrator.ports.labels`	General information about the ports on this Orchestrator. This metric is available in: ■ R82 and higher	Gauge	" "	{'name', 'Interface Name', 'Type', 'Security Group', 'Operating Speed', 'QSFP Mode', 'MTU', 'logical_port', 'port_id', 'auto_ negotiation'}	■ `name` The name of the port on the Orchestrator. ■ `Interface Name` The name of the interface (in Gaia OS) on the Orchestrator. ■ `type` The type (role) of the port on the Orchestrator ("`Mgmt`", "`Uplink`", "`Downlink`", "`SSM Sync`" (internal sync), or "`Site Sync`" (external sync)). ■ `Security Group`

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
					The ID of the Security Group, to which this port is assigned. ■ `Operating Speed` The port speed. ■ `QSFP Mode` The QSFP Mode of the port. ■ `MTU` The port MTU. ■ `logical_port` The ID of the logical port on the Orchestrator (number "<Port Label>" from the physical port ID). ■ `port_id` The internal port ID.

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
					■ `auto_negotiation` The auto-negotiation status ("`Enabled`" or "`Disabled`").
`orchestrator.ports.link_state`	The link state of the ports on this Orchestrator. This metric is available in: ■ R82 and higher	Gauge	{state}	{'name', 'logical_port'}	■ `name` The name of the port on the Orchestrator. ■ `logical_port` The ID of the logical port on the Orchestrator (number "<Port Label>" from the physical port ID).

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`orchestrator.ports.transceiver_state`	The transceiver state in the port on this Orchestrator: ■ 0.0 - Unplugged ■ 1.0 - Plugged (default) This metric is available in: ■ R82 and higher	Gauge	{state}	{'name', 'logical_port'}	■ `name` The name of the port on the Orchestrator. ■ `logical_port` The ID of the logical port on the Orchestrator (number "<Port Label>" from the physical port ID).

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`orchestrator.ports.rx_bytes_per_second`	Received (RX) traffic by the port, in Bytes per second. This metric is available in: ■ R82 and higher	Gauge	{bytes}/s	{'name', 'logical_port'}	■ `name` The name of the port on the Orchestrator. ■ `logical_port` The ID of the logical port on the Orchestrator (number "<Port Label>" from the physical port ID).

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`orchestrator.ports.rx_frames_per_second`	Received (RX) traffic by the port, in packets (frames) per second. This metric is available in: ■ R82 and higher	Gauge	{packets}/s	{'name', 'logical_port'}	■ `name` The name of the port on the Orchestrator. ■ `logical_port` The ID of the logical port on the Orchestrator (number "<Port Label>" from the physical port ID).

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`orchestrator.ports.rx_mbit_per_second`	Received (RX) traffic by the port, in Megabits per second. This metric is available in: ■ R82 and higher	Gauge	Mbit/s	{'name', 'logical_port'}	■ `name` The name of the port on the Orchestrator. ■ `logical_port` The ID of the logical port on the Orchestrator (number "<Port Label>" from the physical port ID).

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`orchestrator.ports.tx_bytes_per_second`	Transmitted (TX) traffic by the port, in Bytes per second. This metric is available in: ■ R82 and higher	Gauge	{bytes}/s	{'name', 'logical_port'}	■ `name` The name of the port on the Orchestrator. ■ `logical_port` The ID of the logical port on the Orchestrator (number "<Port Label>" from the physical port ID).

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`orchestrator.ports.tx_frames_per_second`	Transmitted (TX) traffic by the port, in packets (frames) per second. This metric is available in: ■ R82 and higher	Gauge	{packets}/s	{'name', 'logical_port'}	■ `name` The name of the port on the Orchestrator. ■ `logical_port` The ID of the logical port on the Orchestrator (number "<Port Label>" from the physical port ID).

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`orchestrator.ports.tx_mbit_per_second`	Transmitted (TX) traffic by the port, in Megabits per second. This metric is available in: ■ R82 and higher	Gauge	Mbit/s	{'name', 'logical_port'}	■ `name` The name of the port on the Orchestrator. ■ `logical_port` The ID of the logical port on the Orchestrator (number "<Port Label>" from the physical port ID).

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
`orchestrator.sg_lag`	The current states of Bond (LAG) interfaces for communication between the Orchestrator and the Security Appliances. This metric is available in: ■ R82 and higher	AsyncUpDownCounter	{state}	{'status', 'lag_id', 'lag_id_full', 'gw', 'lag_name', 'lsp_state', 'hostname', 'link_state', 'logical_port', 'membership_state', 'physical_port', 'port_id', 'remote_interface', 'serial', 'sg'}	■ `status` The status of the Bond interface ("`UP`" or "`DOWN`"). ■ `lag_id` The LAG ID. ■ `lag_id_full` The LAG full ID. ■ `gw` The ID of the Security Group Member. ■ `lag_name` The LAG name. ■ `lsp_state` The LSP state ("`FORWARD`" or "`DROP`"). ■ `hostname` The hostname of the remote Security Group Member. ■ `link_state`

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
					The status of the Bond interface link ("UP" or "DOWN"). ■ logical_port The logical port ID ("<Port ID>" or "<Split ID>"). ■ membership_state The LAG Membership State ("Active" or "Inactive"). ■ physical_port The physical port ID (in the format "<Orchestrator ID>/<Port ID>/<Split ID>"). ■ port_id

Metric ID	Metric Description	Metric Type	Metric Units	Metric Labels	Metric Label Description
					The internal port ID. ■ `remote_interface` The name of the peer interface. ■ `serial` The serial number of the Security Appliance. ■ `sg` The ID of the Security Group.

Custom Metrics

Overview

You can configure custom metrics and query them with OpenTelemetry.

This feature is available in:

- R82 and higher
- R81.20 Jumbo Hotfix Accumulator, Take 54 and higher
- R81.10 Jumbo Hotfix Accumulator, Take 135 and higher
- R81 Jumbo Hotfix Accumulator, Take 99 and higher

Procedure

1. Connect to the command line on the Check Point server.
2. Log in to the Expert mode.
3. Create a shell script to configure the required objects for your custom metrics.

   Create this script file in any directory.

   Example: `/home/admin/script_with_custom_metrics.sh`

   Notes:
   - You can use this script file as a template:

     ```
     /opt/CPotlpAgent/custom_
     scripts_template.bash
     ```
   - To create a script file, run:

     ```
     vi /path/to/your/script_
     file
     ```

   This shell script file must contain the lines described below.

   Part 1 (mandatory) - The call to the shell and to the required Check Point script

   Syntax:

   ```
   #!/bin/bash
   . /opt/CPotlpAgent/cs_data_handler_is.bash
   ```

Part 2 (mandatory) - Configure a custom metric object

Syntax:

```
set_ot_object new value <Metric Value>
set_ot_object last label <Metric Label Key 1> <Metric Label
Value 1>
set_ot_object last label <Metric Label Key 2> <Metric Label
Value 2>
...
set_ot_object last label <Metric Label Key N> <Metric Label
Value N>
```

Example:

```
set_ot_object new value 3.1
set_ot_object last label Client_IP 192.168.30.1
set_ot_object new value 4.0
set_ot_object last label Client_IP 192.168.30.2
set_ot_object last label Client_Type SecurityGateway
```

Result JSON array of custom metrics:

```
[
  {
    "value": 3.1,
    "labels": [
      {
        "Client_IP": "192.168.30.1"
      }
    ]
  },
  {
    "value": 4,
    "labels": [
      {
        "Client_IP": "192.168.30.2"
      },
      {
        "Client_Type": "SecurityGateway"
      }
    ]
  }
]
```

Part 3 (optional) - Print verbose output

You can configure a desired log message that your shell script will save in the `/var/log/<script_name>.log` file.

Example:

- If the script file is: `/home/admin/script_with_custom_metrics.sh`
- Then the log file is: `/var/log/script_with_custom_metrics.log`

The line must start with the function name "`verbose_print`" and supports two parameters - a text string "`%s`" and an integer value "`%d`".

Possible positional arguments:

```
verbose_print "%s %d" "<Desired Text Message>" <Desired Integer Value>
```

```
verbose_print "%d %s" <Desired Integer Value> "<Desired Text Message>"
```

Example:

```
verbose_print "My verbose message: %s ; My integer value: %d" "Script executed" 10
```

```
verbose_print "My integer value: %d ; My verbose message: %s" 10 "Script executed"
```

Part 4 (mandatory) - Show an exit message and generate an exit code

Syntax:

```
script_exit "<Desired Exit Message>" <Integer Exit Code>
```

Configure one of these exit codes:

- 0 for success.
- 1 for input error or non-critical error.
- 2 for critical error.

Example:

```
script_exit "Finished running" 0
```

4. Prepare the required payload in a JSON file for your custom metric.

   Create this JSON file in any directory.

Example: `/home/admin/payload_with_custom_metrics.json`

This JSON file must contain these lines (replace the placeholder strings "<...>" with your actual strings):

```
{
  "state" : "<State>",
  "command" : "<Command>",
  "desc" : "<Description>",
  "name" : "<Name>",
  "type" : "<Type>",
  "unit" : "<Unit>",
  "interval" : <Interval>
}
```

Where:

Parameter	Description
`"state"`	The state of the collector: ■ `"enabled"` ■ `"disabled"`
`"command"`	The command that the Skyline Agent needs to run to get the values and labels of the custom metric. The collected data will be in the JSON format - a batch of metric values (real value and label values).
`"desc"`	A description of this custom metric.
`"name"`	The name of the custom metric. Must be unique. Must contain only lowercase Latin letters (`a-z`). Use the period character "`.`" as a separator between each name part. Example: `system.cpu.util` **Best Practice** - For the metric convention name, refer to https://opentelemetry.io/docs/specs/semconv/general/metrics/ .

Parameter	Description
`"type"`	Can be one of these: ■ `"Counter"` To count something by recording a delta value. • If the value is monotonic, use `"Counter"`. • Otherwise, use `"UpDownCounter"`. ■ `"UpDownCounter"` To measure something by recording an absolute value that is additive / summable. ■ `"Gauge"` To measure something by recording an absolute value that is **not** additive / summable. **Best Practice** - For the metric types, refer to https://uptrace.dev/opentelemetry/metrics.html.
`"unit"`	The measurement unit for the custom metric. For example, the unit for the metric "`hardware.memory.util`" is "`bytes`". If the custom metric uses a non-standard unit, you must enclose it in curly braces `{ }`. For example : "`{interval}`". **Best Practice** - For the unit convention, refer to https://github.com/open-telemetry/semantic-conventions/blob/main/docs/general/metrics.md#instrument-units.
`"interval"`	How frequently (in seconds) the Skyline Agent needs to run the command.

Example:

```
{
  "state" : "enabled",
  "command" : "cat /home/admin/HW_sensors_data.json",
  "desc" : "The current temperature in the appliance",
  "name" : "hardware.sensors.temperature",
  "type" : "Gauge",
  "unit" : "{celcius}",
  "interval" : 2
}
```

5. Run the configuration command to apply the JSON payload:

```
sklnctl otlp add -name /<Path>/<Shell Script File> --path
/<Path>/<JSON File>
```

Example:

```
sklnctl otlp add -name /home/admin/script_with_custom_
metrics.sh --path /home/admin/payload_with_custom_metrics.json
```

6. Restart the OpenTelemetry Agent:

```
/opt/CPotlpAgent/CPotlpagentCli.sh stop
/opt/CPotlpAgent/CPotlpagentCli.sh start
```

7. Examine the currently exported metrics:

```
sklnctl otelcol metrics --show
```

Notes

- During an upgrade, the Gaia OS automatically moves your shell script to this directory:

  `/var/log/CPotlpAgent/backup/scripts`

- If there are issues with your custom metrics, then:

  1. Restart the OpenTelemetry Agent:

     ```
     /opt/CPotlpAgent/CPotlpagentCli.sh stop
     /opt/CPotlpAgent/CPotlpagentCli.sh start
     ```

  2. Examine these log files:

     - `/opt/CPotelcol/sklnctl.log`
     - `/opt/CPotlpAgent/otlp_agent.log`