Monitoring Guide

VeloCloud SD-WAN Gateway

Version 6.1

Arista.com

Headquarters	Support	Sales
5453 Great America Parkway Santa Clara, CA 95054 USA +1-408-547-5500	+1-408-547-5502 +1-866-476-0000	+1-408-547-5501 +1-866-497-0000
www.arista.com/en/	support@arista.com	sales@arista.com

Contents

Chapter 1

VeloCloud SD-WAN Gateway Monitoring Guide

The VeloCloud Gateway Monitoring Guide discusses the core components of the VeloCloud Gateways and explains how to monitor your own Gateway deployments.

Version Compatibility

While the principles in this guide are largely applicable to any version of the Gateway, certain commands and example outputs may only be relevant to Release 5.0.0.

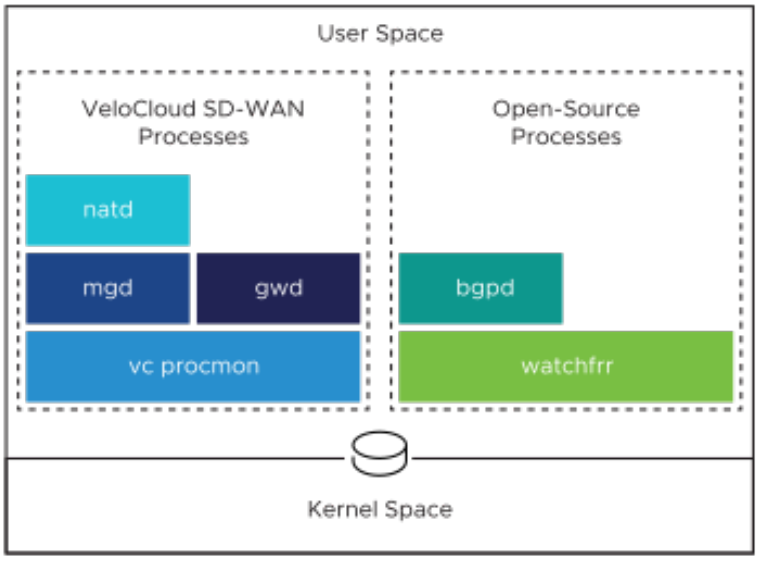
Chapter 2

Components

The Gateway runs on an Ubuntu Operating System version 18.04. However, traditional Open Source components such as the Linux routing and firewall (iptables) subsystem are not involved in processing the user traffic.

The entire networking stack is implemented in the user space in a process called gwd. While gwd contains the vast majority of functionality in the system, there are various other components that support the operation of the Gateway.

Figure 2-1: Network Stack



The following table lists the processes with description and log files.

Table 1: Processes with Description and Log Files

Process	Description	Log File
vc_procmon	VeloCloud SD-WAN Process Monitor (vc_procmon) is the foundational process of the VeloCloud SD-WAN system. This process is responsible for launching other VeloCloud SD-WAN processes, re-launching them on failure, and monitoring memory consumption of gwd.	**/var/log/vc_procmon.log**
mgd	The Management Plane Daemon (mgd) is responsible for communication with the Orchestrator. This process is kept isolated from gwd so that in the incident of a total failure of the gwd process, the Orchestrator is still reachable for configuration changes or software updates required to resolve the failure.	**/var/log/mgd.log**
gwd	This process comprises the entire Data and Control Plane of the Gateway (except for dynamic routing protocols like BGP). Use **debug.py** and **dispcnt** to query about the process.	**/var/log/gwd.log**
natd	This process manages the assignment of Port Address Translation (PAT) entries and stores them in shared memory, ensuring that the same NAT translations are done even after gwd is restarted or upgraded. Use **debug.py** to query about the process.	**/var/log/natd.log**
watchfrr	The watchfrr daemon is part of the **FRR** open source routing library, and is responsible for launching bgpd, re-launching it on failure, and running any other related utilities. The script `/usr/sbin/frr.init` which is available on Gateway, can be used to restart some daemons.	N/A
bgpd	The BGP Daemon (bgpd) is part of the **FRR** open source routing library and manages the BGP neighbors and routes.	**/var/log/bgpd.log**
bfdd	The BFD Daemon (bfdd) is part of the **FRR** open source routing library which is used to detect route failures between two connected entities faster with low-overhead detection of failures.	**/var/log/bfdd.log**

The **gwd1** is an interface, which provides the ability for gwd to deliver packets to the kernel. An example is a packet destined for the local gateway host but received by gwd.

Chapter 3

Cached Configuration

Use the cached configuration on Gateways to check the data received from the Orchestrator.

The configuration is cached in the following file: **/opt/vc/.gateway.info**.

Use the script **/opt/vc/bin/getpolicy.py** to read the cached configuration and to verify the data received from the Orchestrator.

```
vcadmin@vcg1-example:~$ /opt/vc/bin/getpolicy.py managementPlane { "schemaVersion": "1.7.0",
 "version": "0", "data": { "heartBeatSeconds": 30, "managementPlaneProxy": { "primary": "vco1-
example.velocloud.net", "secondary": null }, "timeSliceSeconds": 300, "vcoAddress": "1.2.3.4",
 "statsUploadSeconds": 300 }, "module": "managementPlane" } vcadmin@vcg1-example:~$
```

An exception to the above is Control Plane configuration. As the Control Plane module contains sensitive information such as pre-shared keys for the Non SD-WAN Destination tunnels, it is never cached locally. Hence, if the Gateway restarts while not connected to the Orchestrator, the components and processes that require a Control Plane blob like Non SD-WAN Destinations, VLAN/VRF handoff, and BGP routing do not work until the Orchestrator connectivity is restored.

As an exception, when the Gateway loses connection to Orchestrator and does not reboot all the Control Plane modules will function normally.

Chapter 4

Monitor Gateways on Edge Cloud Orchestrator

Operator and Partners can have a high-level view on the status of the Gateway from the Orchestrator.

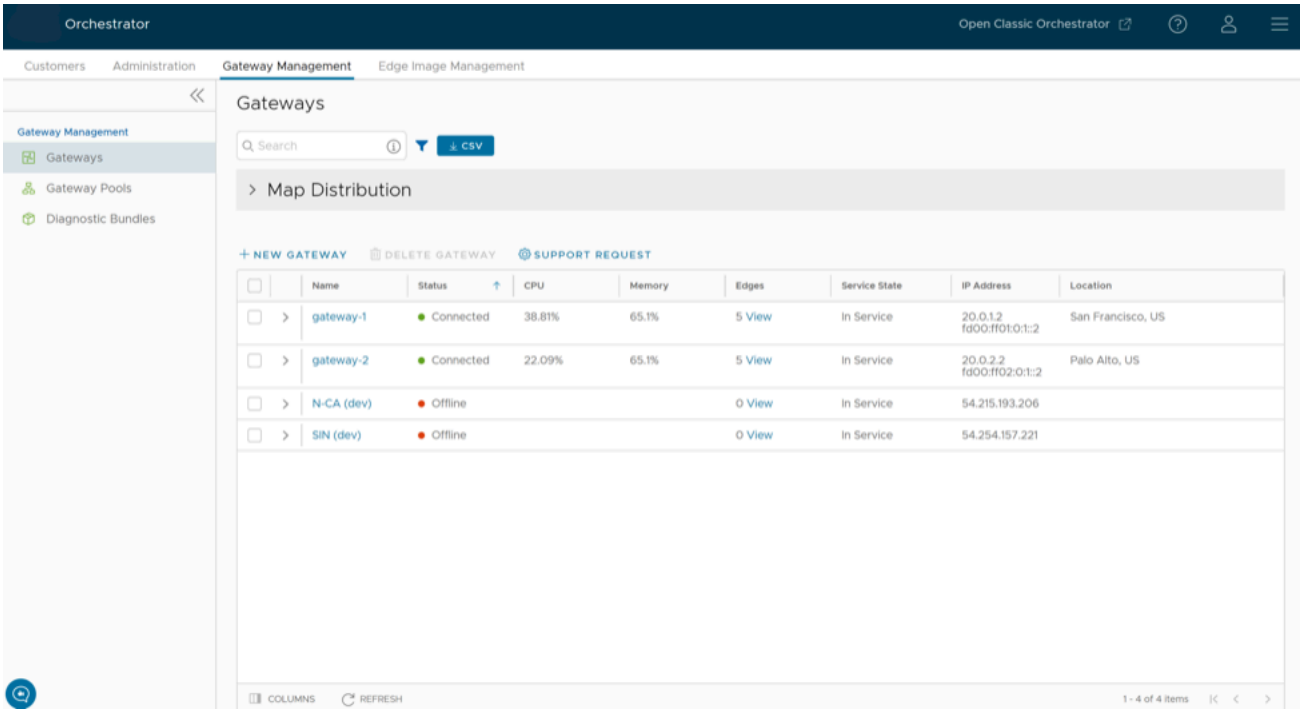
4.1 Monitor Gateways

You can monitor the status and network usage data of Gateways available in the Operator and the Partner portal.

To monitor the Gateways:

1. In the **Operator** portal, select **Gateway Management** > **Gateways**.
2. The **Gateways** page displays the list of available Gateways.

   Figure 4-1: Gateways Page



3. Select **Map Distribution** to expand and view the locations of the Gateways in the Map. By default, this view is collapsed.
4. You can also select the arrows prior to each Gateways name to view additional details.

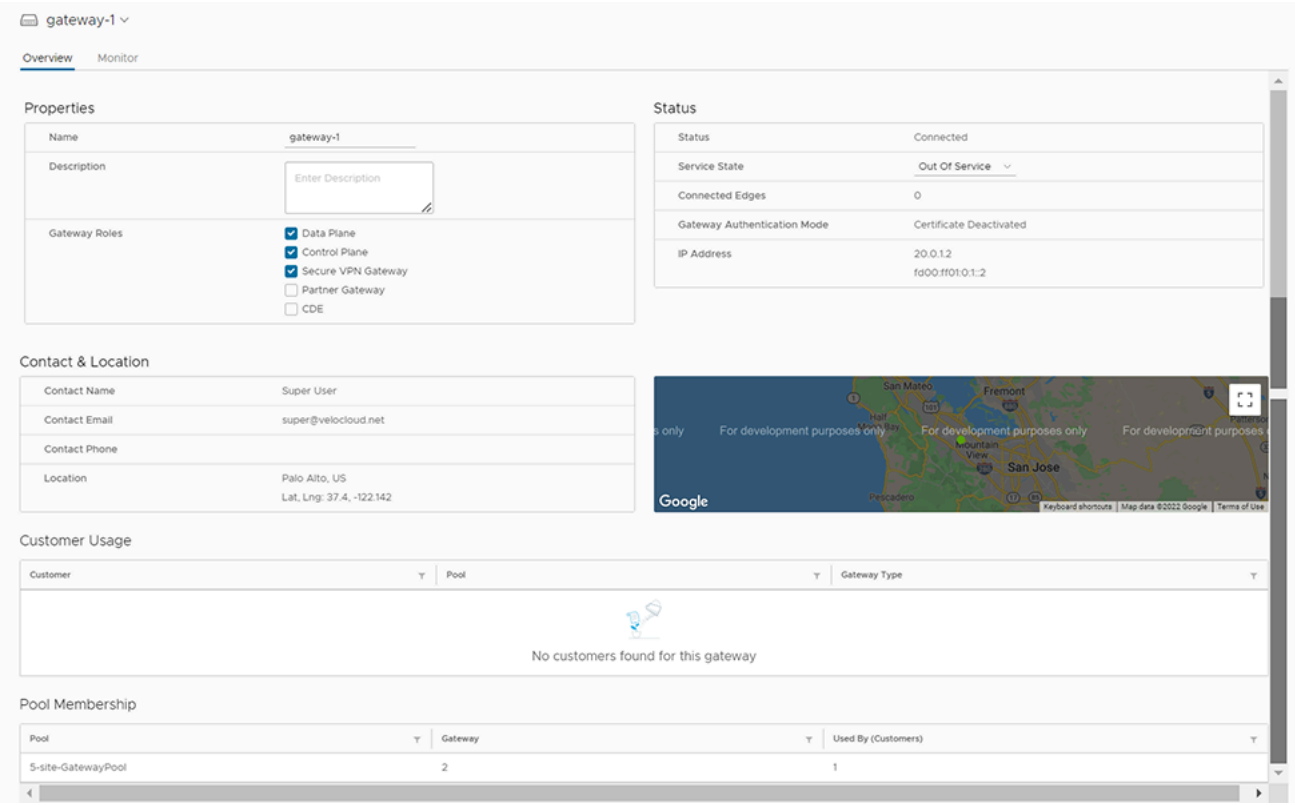
   The page displays the following details:

   - **Name** – Name of the Gateways.

- **Status** – Current status of the Gateways. The status may be one of the following: Connected, Degraded, Never Activated, Not in Use, Offline, Out of Service, or Quiesced.
- **CPU** – Percentage of CPU utilization by the Gateways.
- **Memory** – Percentage of memory utilization by the Gateways.
- **Edges** – Number of Edges connected to the Gateways.
- **Service State** – Service state of the Gateways. The state may be one of the following: Historical, In Service, Out of Service, Pending Service, or Quiesced.
- **IP Address** – The IP Address of the Gateways.
- **Location** – Location of the Gateways.

5. In the Search field, enter a term to search for specific details. Select the **Filter** icon to filter the view by a specific criterion.
6. Select the **CSV** option to download a report of the Gateways in the CSV format.
7. Select the link to a Gateway to view the details of the selected Gateway.

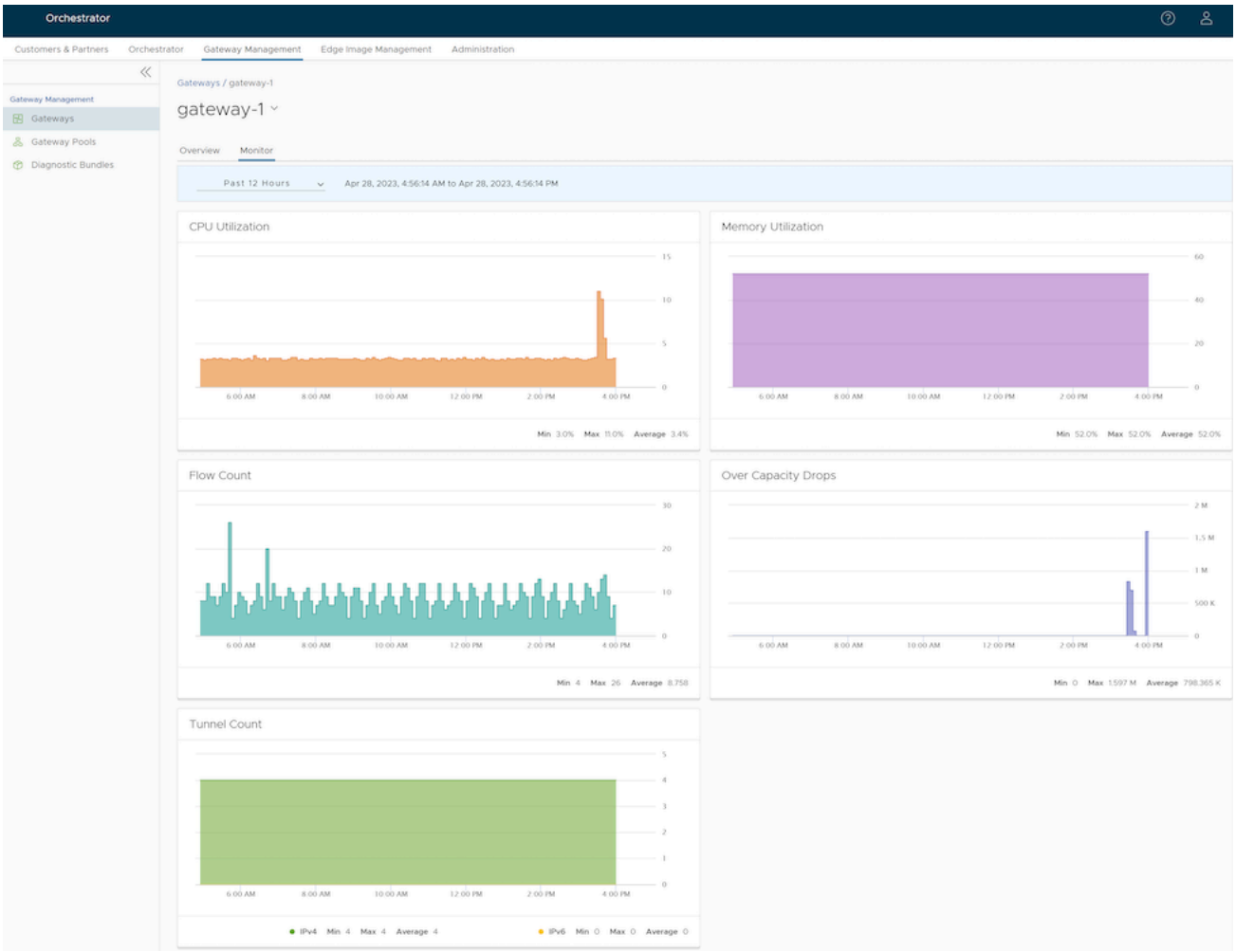
Figure 4-2: Selected Gateway Details



The **Overview** tab displays the properties, status, location, customer usage, and Gateway Pool of the selected Gateway.

8. Select the **Monitor** tab to view the usage details of the selected Gateways.

Figure 4-3: Monitoring Usage Details of the Gateways



At the top of the page, you can choose a specific time period to view the details of the Gateway for the selected duration.

The page displays graphical representation of usage details of the following parameters for the period of selected time duration, along with the minimum, maximum, and average values.

- **CPU Percentage** – Percentage of usage of CPU.
- **Memory Usage** – Percentage of usage of memory.
- **Flow Counts** – Count of traffic flow.
- **Over Capacity Drops** – Total number of packets dropped due to over capacity since the last sync interval. Occasional drops are expected, usually caused by a large burst of traffic. However, a consistent increase in drops usually indicates a Gateway capacity issue.
- **Tunnel Count** – Count of tunnel sessions for both the IPv4 and IPv6 addresses.

Hover the mouse on the graphs to view additional details.

Note: If you are a Partner user, follow the above instructions in the Partner portal to view the details of the Partner Gateways.

Chapter 5

Monitor Gateways using CLI

You can use CLI commands to check the status of Gateways in an Orchestrator.

5.1 Monitor System Health

You can use the CLI commands to check the status of the Gateways, Software version, usage of CPU and memory, and other information.

5.1.1 Monitor Gateway Activation State

Use the following command to check whether the Gateway is activated on an Orchestrator.

In the following example, the Gateway is activated:

```
vcadmin@vcg1-example1:~$ /opt/vc/bin/is_activated.py True vcadmin@vcg1-example1:~$
```

In the following example, the Gateway is deactivated:

```
vcadmin@vcg1-example1:~$ /opt/vc/bin/is_activated.py False vcadmin@vcg1-example1:~$
```

5.1.2 View Activated Orchestrator Name

Use the following command to find out the Orchestrator to which the Gateway belongs to, provided the Gateway is activated.

```
vcadmin@vcg1-example1:~$ /opt/vc/bin/getpolicy.py managementPlane.data.managementPlanePro
xy.primary "vco1-example1.velocloud.net" vcadmin@vcg1-example1:~$
```

5.1.3 View Software Version

The various processes in the system display their own version numbers, which should all be the same.

Check the version numbers using the following commands:

```
root@NY-GATEWAY-1:~# /opt/vc/sbin/gwd -v VCG Info ======== Version: 4.2.0 Build rev: R420-20201216
-GA-0bcea3f6f0 Build Date: 2020-12-16_23-23-33 Build Hash: 0bcea3f6f0e6b8c21260187bb2d953e4cefd7f2
7
```

```
root@NY-GATEWAY-1:~# /opt/vc/sbin/natd -v NATd Info ======== Version: 4.2.0 Build rev:
 R420-20201216-GA-0bcea3f6f0 Build Date: 2020-12-16_23-23-33 Build Hash: 0bcea3f6f0e6b
8c21260187bb2d953e4cefd7f27
```

```
root@NY-GATEWAY-1:~# /opt/vc/sbin/mgd -v VeloCloud gateway 4.2.0 build R420-20201216-GA-0bcea3f6f0
```

5.1.4 View NTP Time Zone

Use the following command to view the NTP time zone. The Gateway time zone must be set to **Etc/UTC**.

```
vcadmin@vcg1-example:~$ cat /etc/timezone Etc/UTC vcadmin@vcg1-example:~$
```

If the time zone is incorrect, use the following commands to update the time zone.

```
echo "Etc/UTC" | sudo tee /etc/timezone sudo dpkg-reconfigure --frontend noninteractive tzdata
```

5.1.5 View NTP Offset

Use the following command to view the NTP offset, which must be less than or equal to 15 milliseconds.

```
sudo ntpqvcadmin@vcg1-example:~$ sudo ntpq -p remote refid st t when poll reach delay offset
 jitter ============================================================================== *ntp1-
us1.prod.v 74.120.81.219 3 u 474 1024 377 10.171 -1.183 1.033 ntp1-eu1-old.pr .INIT. 16 u - 1024 0
 0.000 0.000 0.000 vcadmin@vcg1-example:~$
```

If the offset is incorrect, use the following commands to update the NTP offset.

```
sudo systemctl stop ntp sudo ntpdate <server> sudo systemctl start ntp
```

5.1.6 Monitor Disk Usage

Use the following command to check the disk usage space. Ensure that the disk has at least 16 GB of free space to store critical files like logs and cores.

```
vcadmin@vcg1-example:~$ sudo df -kh --total | grep total | awk '{print $4}' 77G vcadmin@vcg1-
example:~$
```

The common places for disk usage to build up are **`/var/log`**, **`/velocloud/core`**, and **`/tmp`**.

Note: Each session creates a temporary file named as **`/velocloud/vctcpdump.XXXX`** with fixed size of 1MB. This file is deleted when the session ends.

5.1.7 Monitor CPU Usage

The Gateway processes bursts of traffic and bursts of high CPU are expected. The Gateway should be monitored for CPU cores that are spinning at 100%. However, the DPDK cores run in poll mode for performance reasons and they expect to take close to 100% CPU at high throughput.

You can monitor a Gateway with thresholds that provide warning or critical states which indicate potential issues prior to impacting services. The following table lists the threshold values and recommended actions.

Table 2: CPU Usage Threshold

Threshold State	Threshold Value		Recommended Corrective Action
	DP Core	Non DP Core	
Warning	95%	80%	If the threshold value is crossed consistently for 5 minutes: • Check per-process CPU usage. • Monitor for 10 more minutes. If the threshold value is crossed consistently for 5 minutes: • Collect Gateway diagnostic bundle. • Open a support case.
Critical	98%	90%	If the threshold value is crossed consistently for 5 minutes: • Monitor for possible critical packet drop which can indicate over capacity. If the issue is observed for one hour: • If over capacity is observed over a five minute interval, add Gateway capacity and rebalance to avoid capacity related service impact. **Note:** Before rebalancing the Gateway, confirm that the capacity metrics are within the recommended limit. For additional information on capacity metrics, see Capacity of Gateway Components.

The following is an example Python script for monitoring the CPU usage:

Note: You can also use Telegraf to monitor the CPU usage. For additional information, see Monitor Gateways using Telegraf.

```
#! /usr/bin/env python """ Check for CPUs spinning at 100% """ import re import collections import
 time import sys import json import os import subprocess re_cpu = re.compile(r"^cpu\d+\s") CPUStat
 = collections.namedtuple('CPUStat', ['user', 'nice', 'sys', 'idle']) def get_stats(): stats
 = open("/proc/stat").readlines() ret = {} for s in stats: if not re_cpu.search(s): continue
 s = s.split() ret[s[0]] = CPUStat(*[ int(v) for v in s[1:5]]) return ret def verify_dpdk_s
upport(): if os.path.isfile('/opt/vc/etc/dpdk.json'): with open("/opt/vc/etc/dpdk.json") as
 data: d=json.loads((data.read())) if "status" in d.keys(): return True if d['status'] is
 "Supported" else False else: return False def another_verify_dpdk_support(): if os.path.isfil
```

```
e('/opt/vc/bin/debug.py'): f=subprocess.check_output(["/opt/vc/bin/debug.py","--dpdk_ports_d
ump"]) x=[r.split() for r in f.split('\n')] if len(x) <= 1: return False else: return True else:
 return False dpdk_status=verify_dpdk_support() or another_verify_dpdk_support() if __name__ ==
 "__main__": try: stat1 = get_stats() time.sleep(3) stat2 = get_stats() except: print "UKNOWN
 - failed to get CPU stat: %s" % str(sys.exc_info()[1]) sys.exit(3) busy_cpu_set = [ cpu for
 cpu in stat1 if (stat2[cpu].idle - stat1[cpu].idle)==0 ] if not busy_cpu_set: print "OK - no
 spinning CPUs" sys.exit(0) if dpdk_status == True: if "cpu1" in busy_cpu_set and len(busy_cpu_
set) == 1: print "OK - no spinning CPUs" sys.exit(0) elif "cpu1" in busy_cpu_set: busy_cpu_set.
remove('cpu1') print "CRITICAL - %s is at 100%%" % (",".join(busy_cpu_set)) sys.exit(2) else:
 print busy_cpu_set,1 print "CRITICAL - %s is at 100%%" % (",".join(busy_cpu_set)) sys.exit(2)
 else: print "CRITICAL - %s is at 100%%" % (",".join(busy_cpu_set)) sys.exit(2)
```

5.1.8 Monitor Memory Usage

The main process (gwd) has its memory monitored by **`vc_process_monitor`**, which ensures that it never consumes more than 75% of available memory. As a result, monitoring for total system memory is done with a warning threshold of 80% and critical threshold of 90%.

You can monitor a Gateway with thresholds that provide warning or critical states which indicate potential issues prior to impacting services. The following table lists the threshold values and recommended actions.

Table 3: Threshold Values and Recommended Actions

Threshold State	Threshold Value	Recommended Corrective Action
Warning	80%	If the memory crosses warning threshold: • Collect Gateway diagnostic bundle. • Check per-process memory usage Continue monitoring actively and check for increasing utilization.
Critical	90%	If the memory crosses critical threshold: • Monitor for possible critical packet drop which can indicate over capacity. If the issue is observed again: • If over capacity is observed over a 5 minute interval, add Gateway capacity and rebalance to avoid capacity related service impact. **Note:** Before rebalancing the Gateway, confirm that the capacity metrics are within the recommended limit. For additional information on capacity metrics, see Capacity of Gateway Components.

The following is an example Python script for monitoring the memory usage:

Note: You can also use Telegraf to monitor the memory usage. For additional information, see Monitor Gateways using Telegraf.

```
#!/usr/bin/env python from optparse import OptionParser import sys # Parse commandline options:
 parser = OptionParser(usage="%prog -w <warning threshold>% -c <critical threshold>% [ -h ]")
 parser.add_option("-w", "--warning", action="store", type="string", dest="warn_threshold",
 help="Warning threshold in absolute(MB) or percentage") parser.add_option("-c", "--critical",
 action="store", type="string", dest="crit_threshold", help="Critical threshold in ansolute(MB)
 or percentage") (options, args) = parser.parse_args() def read_meminfo(): meminfo = {} for line
 in open('/proc/meminfo'): if not line: continue (name, value) = line.split()[0:2] meminfo[name.
strip().rstrip(':')] = int(value) return meminfo if __name__ == '__main__': if not options.crit_
```

```
threshold: print "UNKNOWN: Missing critical threshold value." sys.exit(3) if not options.warn_
threshold: print "UNKNOWN: Missing warning threshold value." sys.exit(3) is_warn_pct =
 options.warn_threshold.endswith('%') if is_warn_pct: warn_threshold = int(options.warn_threshold
[0:-1]) else: warn_threshold = int(options.warn_threshold) is_crit_pct = options.crit_
threshold.endswith('%') if is_crit_pct: crit_threshold = int(options.crit_threshold[0:-1]) else:
 crit_threshold = int(options.crit_threshold) if crit_threshold >= warn_threshold: print "UNKNOWN:
 Critical percentage can't be equal to or bigger than warning percentage." sys.exit(3) meminfo
 = read_meminfo() memTotal = meminfo["MemTotal"] memFree = meminfo["MemFree"] + meminfo["Buff
ers"] + meminfo["Cached"] memFreePct = 100.0*memFree/memTotal if (is_crit_pct and memFreePct
 <= crit_threshold) or (not is_crit_pct and memFree/1024<=crit_threshold): print "CRITICAL:
 Free memory is at %2.0f %% ( %d MB free our of %d MB total)" % (memFreePct, memFree/1024,
 memTotal/1024) sys.exit(2) if (is_warn_pct and memFreePct <= warn_threshold) or (not is_warn_pct
 and memFree/1024<=warn_threshold): print "WARNING: Free memory is at %2.0f %% ( %d MB free
 our of %d MB total)" % (memFreePct, memFree/1024, memTotal/1024) sys.exit(1) else: print "OK:
 Free memory is at %2.0f %% ( %d MB free our of %d MB total)" % (memFreePct, memFree/1024,
 memTotal/1024) sys.exit(0)
```

5.2 Monitor VeloCloud SD-WAN Services

Use the CLI commands to monitor the SD-WAN processes, sessions, and components.

5.2.1 Monitor VeloCloud SD-WAN Processes

The VeloCloud SD-WAN processes described in the Components should be running in order to ensure proper functionality of the system. The Linux command **`pgrep`** can be used to identify the process. The format is slightly different for Python processes. If the process is running, a pid (integer process ID) is returned. If it is not running, the output is empty.

vc_procmon

Use the following command to check if **vc_procmon** is running on the system.

```
vcadmin@vcg1-example:~$ pgrep -f vc_procmon 14711 vcadmin@vcg1-example:~$
```

Other Processes

Use the following commands to check for other processes.

```
vcadmin@vcg1-example:~$ pgrep mgd 14725 vcadmin@vcg1-example:~$ pgrep gwd 15143 vcadmin@vcg1-
example:~$ pgrep natd 15095
```

To recover the processes, restart the VeloCloud SD-WAN Process Monitor, which restarts all other processes. Use the following command to restart VeloCloud SD-WAN Process Monitor.

```
sudo service vc_process_monitor restart
```

Use the following command to restart routing protocol daemons.

```
/usr/sbin/frr.init {start|stop|restart} [daemon ...]
```

5.2.2 Monitor Certificate Revocation List

On Gateways with PKI enabled, the revoked certificates are stored in a Certificate Revocation List (CRL). If this list grows too long, generally due to an issue with the Certificate Authority of the Orchestrator, the performance of the Gateway is impacted. The CRL should be less than 4000 entries long.

Use the following command to check the CRL entries.

```
vcadmin@vcg1-example:~$ openssl crl -in /etc/vc-public/vco-ca-crl.pem -text | grep 'Serial Number'
 | wc -l 14 vcadmin@vcg1-example:~$
```

5.2.3 Monitor ICMP Status

If a Gateway is configured as a Partner Gateway with static routing, and the ICMP responder is configured to track the reachability of those routes, a **`debug.py`** command is provided to indicate the UP or DOWN states, as follows:

```
vcadmin@vcg1-example:~$ sudo /opt/vc/bin/debug.py --icmp_monitor { "icmpProbe": { "cTag": 0,
 "destinationIp": "0.0.0.0", "enabled": false, "frequencySeconds": 0, "probeFail": 0, "probeType":
 "NONE", "probesSent": 0, "respRcvd": 0, "sTag": 0, "state": "DOWN", "stateDown": 0, "stateUp":
 0, "threshold": 0 }, "icmpResponder": { "enabled": false, "ipAddress": "0.0.0.0", "mode":
 "CONDITIONAL", "reqRcvd": 0, "respSent": 0, "state": "DOWN" } } vcadmin@vcg1-example:~$
```

When the ICMP responder is enabled, the DOWN state means that there are no Edges connected to the Gateway.

5.2.4 Monitor BGP Sessions

The **`debug.py`** command **`bgp_view_summary`** provides information about state of the BGP neighbor and prefixes learned, to verify that BGP is UP and exchanging prefixes. Use the following command to verify if BGP neighborships are established.

```
vcadmin@vcg1-example:~$ /opt/vc/bin/debug.py --bgp_view_summary | grep Established | wc -l 6
 vcadmin@vcg1-1:~$
```

If the BGP sessions are down, check whether the Gateway is properly connected to the Orchestrator.

5.2.5 Monitor Core Files

Whenever a service crashes on the Gateway, a core file is generated. The diagnostic bundles generated from the Orchestrator should be retrieved as soon as possible following the generation of a core file, to download the core file and to provide the associated logs to Arista Support.

The following example illustrates a Python script to check for recent core files:

```
#! /usr/bin/env python import subprocess, traceback, os, os.path,glob,datetime,time,sys,re
 from pynag.Plugins import PluginHelper,ok,warning,critical,unknown from subprocess import
 Popen, PIPE import time import os import commands import json helper = PluginHelper()
 helper.parse_arguments() def diag_check(): regex_patern = "^.*\s+Uploading diag-201[0-9]
-.*" re_nat = re.compile(regex_patern) cmd = 'grep "Uploading diag-201[0-9]" /var/log/mgd.
log' p1 = subprocess.Popen([cmd], stdout=subprocess.PIPE,stderr=subprocess.PIPE,shell=True)
```

```
 stdout_value, stderr_value = p1.communicate() m = re_nat.search(stdout_value) if m: return
 True else: return False def vco_vcg_version(): with open("/opt/vc/.gateway.info") as data:
 d=json.loads((data.read())) vcg=d["gatewayInfo"]["name"] #build_number=d["gatewayInfo"]
["buildNumber"] status,output = commands.getstatusoutput("sudo /opt/vc/sbin/gwd -v 2>&1 |
 grep rev") if status == 0: build_number=output.split()[2].rstrip('\n') vco=d["configuration"]
["managementPlane"]["data"]["managementPlaneProxy"]["primary"] return vcg,build_number,vco
 status_file = "/tmp/coredump_status_file" warning_file = "/tmp/warning_file" if not os.path.isfil
e(status_file) and not os.access(status_file, os.R_OK): os.system("touch /tmp/coredump
_status_file") os.system("chown nagios:nagios /tmp/coredump_status_file") if not os.path.isfil
e(warning_file) and not os.access(status_file, os.R_OK): os.system("touch /tmp/warning_file")
 os.system("chown nagios:nagios /tmp/warning_file") if not os.path.isfile(warning_file) and
 not os.access(status_file, os.R_OK): os.system("touch /tmp/crashlist.txt") os.system("chown
 nagios:nagios /tmp/crashlist.txt") command = "cat /tmp/coredump_status_file" command1 = "cat /
tmp/warning_file" files = ["crashlist.txt","warning_file","coredump_status_file","coredump_
message"] for item in files: if os.path.isfile("/tmp/"+item): st=os.stat("/tmp/"+item) if
 st.st_uid == 0: commands.getstatusoutput("sudo chown nagios:nagios /tmp/"+item) status,output
 = commands.getstatusoutput(command) if output == "1": status_message = "" os.system("chown
 nagios:nagios /tmp/coredump_message") with open("/tmp/coredump_message", "r") as data: for line
 in data.readlines(): status_message += line mtime = os.path.getmtime("/tmp/coredump_status_
file") cur_time = time.time() if int(cur_time) - int(mtime) >= 300: os.system('echo -n "0"
 > /tmp/coredump_status_file') helper.status(critical) helper.add_summary(status_message)
 helper.exit() sys.exit(0) status_message = "" newcore = 0 try: crashlistpath = '/tmp/crashli
st.txt' cmd = "stat -c '%Y %n' /velocloud/core/*core.tgz" if not os.path.isfile(crashlistpath)
 and not os.access(crashlistpath, os.R_OK): os.system("find /velocloud/core/ -name *core.tgz
 > /tmp/crashlist.txt") with open(crashlistpath, "a+") as f: oldcrashlist = f.read() corelist
 = glob.glob("/velocloud/core/*core.tgz") corecount = len(corelist) if corecount > 0 : for
 line in corelist: file_modified = datetime.datetime.fromtimestamp(os.path.getmtime(line)) if
 datetime.datetime.now() - file_modified > datetime.timedelta(hours=42*24): os.remove(line) if
 not line in oldcrashlist: newcore +=1 status_message += '\n' + "Core:" +str(newcore) +" " +
 line.rsplit('/',1)[1] + " " f.write(line+'\n') cmd1 = "tar -xvf " + line.rstrip('\n') + " -C /
tmp --wildcards --no-anchored '*.txt' " crash = subprocess.Popen(cmd1, shell=True, stdout=subpro
cess.PIPE) crash.wait() for line1 in crash.stdout: btcmd = "awk '/^Thread 1 /,/^----/' /tmp/" +
 line1.rstrip('\n') + " | egrep '^#' | sed 's/ 0x0.* in //' | sed 's/ (.*/ /'" bt = subprocess.Po
pen(btcmd, shell=True, stdout=subprocess.PIPE) status_message += '\n'+ bt.communicate()[0] else:
 helper.status(ok) status_message = "No Core file" f.close() except Exception as e: traceback.pri
nt_exc() helper.exit(summary="Nagios check could not complete", long_output=str(e), exit_code=unk
nown, perfdata='') if corecount and not newcore: helper.status(ok) status_message = str(corecount
)+ " old core file found in /velocloud/core" os.system('echo -n "0" > /tmp/coredump_status_file'
) elif newcore > 0: output = vco_vcg_version() vcg_data = "%s; VCG_Build_Number:%s; VCO:%s
\n" %(output) status_message = vcg_data + str(newcore)+ " New Core\n"+ status_message with
 open("/tmp/coredump_message", "w") as data: data.writelines(status_message) os.system('echo -
n "1" > /tmp/warning_file') os.system('echo -n "1" > /tmp/coredump_status_file') helper.status
(critical) helper.add_summary(status_message) helper.exit() sys.exit(0) status,output_warn =
 commands.getstatusoutput(command1) if output_warn == "1" : helper.status(warning) status_message
 = "Please generate gateway diag bundle from the VCO if required" result = diag_check() if result
 == False: if not os.path.isfile("/tmp/coredump_start_time"): os.system("touch /tmp/coredump
_start_time") os.system("chown nagios:nagios /tmp/coredump_start_time") start_time = time.time()
 with open("/tmp/coredump_start_time", "w") as data: data.write(str(start_time)) end_time =
 time.time() cmd = "cat /tmp/coredump_start_time" status,start_time = commands.getstatusoutput(c
md) total_time = end_time - float(start_time) if total_time > 10800: result = True if result
 == True: os.system('echo -n "0" > /tmp/warning_file') os.remove ("/tmp/coredump_start_time")
 helper.status(warning) status_message = "Please generate the diagbundle for the last crash. if it
 is taken already, please ignore this message" helper.add_summary(status_message) helper.exit()
```

5.3 Capacity of Gateway Components

Use the CLI commands to check the capacity of the Gateway components and verify that the configured values do not exceed the supported values, to ensure seamless performance.

For supported values, refer to the https://www.arista.com/en/support/product-documentation.

5.3.1 Monitor Packet Processing Queue

The packet processing engine on the Gateway involves multiple stages, and each stage has a packet processing queue in between. Due to the bursty nature of traffic through a Gateway, occasional packet builds up in the packet forwarding queues are expected. However, consistent high queue length in certain queues indicate a capacity problem.

The following example shows output of the **debug.py** command to view handoff queue output.

The output has been truncated to display only the first and last entries, for conciseness. You can exclude the **-v** option in the command to view the output in tabular format.

```
vcadmin@vcg1-example:~$ /opt/vc/bin/debug.py -v --handoff { "handoffq": [ { "deq": 12126489784,
 "drops": 0, "enq": 12126482089, "name": "vc_queue_net_sch", "qlength": 0, "qlimit": 4096,
 "sleeping": 1475174572, "tid": 1502, "wmark": 1280, "wmark_1min": 385, "wmark_5min": 450,
 "wokenup": 1164664965 }, … { "deq": 767292, "drops": 0, "enq": 767272, "name": "vc_queue_ip_
common_bh_1", "qlength": 0, "qlimit": 16384, "sleeping": 596612, "tid": 1512, "wmark": 53,
 "wmark_1min": 3, "wmark_5min": 3, "wokenup": 596209 } ] } vcadmin@vcg1-example:~$
```

You need to note the values of **qlength** and **wmark**.

The **qlength** column indicates the number of packets that are currently buffered in the queue. The **wmark** column indicates the maximum depth a queue has ever reached, which indicates how close a Gateway has come to dropping packets. The impact and remediation for these depends largely on the queue being monitored.

You should monitor both the critical and non-critical queues.

5.3.1.1 Netif and Per-Core Queues

The high watermark level in these queues indicates that the packet processing rate is lower than the incoming rate.

The following example shows output of the **dispcnt -p netif -s len -s wmark -d vcgw.com** command.

```
# dispcnt -p netif -s len -s wmark -d vcgw.com Wed Jul 20 09:03:39 2022 netif_queue0_len = 0
 0 /s netif_queue0_wmark = 0 0 /s netif_queue0_wmark_1min = 0 0 /s netif_queue0_wmark_1s = 0
 0 /s netif_queue0_wmark_5min = 0 0 /s netif_queue1_len = 0 0 /s netif_queue1_wmark = 45 22 /s
 netif_queue1_wmark_1min = 3 1 /s netif_queue1_wmark_1s = 1 0 /s netif_queue1_wmark_5min = 3 1 /s
```

In the output, the value in the first column is the current count, and the second column is the rate per second. When you run this command for the first iteration, the first column displays the total (lifetime) count since the counter started, and the second column displays the rate which is calculated by dividing the lifetime total by 2. As the data refreshes every 2 seconds, then the first column displays the count since the last sample, and the second column is the rate per second.

The following example shows output of the **dispcnt -p per_core -s len -s wmark -d vcgw.com** command.

```
# dispcnt -p per_core -s len -s wmark -d vcgw.com Wed Jul 20 09:11:05 2022 per_core_queue0_len =
 0 0 /s per_core_queue0_wmark = 1476 738 /s per_core_queue0_wmark_1min = 91 45 /s per_core_queu
e0_wmark_1s = 34 17 /s per_core_queue0_wmark_5min = 346 173 /s per_core_queue1_len = 0 0 /
s per_core_queue1_wmark = 1216 608 /s per_core_queue1_wmark_1min = 47 23 /s per_core_queu
e1_wmark_1s = 27 13 /s per_core_queue1_wmark_5min = 64 32 /s
```

5.3.1.2 View Non-Critical Queues

The high queue length in the non-critical queues are less common or less likely to impact customers.

The following are the non-critical queues that can be monitored.

vc_queue_vcmp_init – This queue provides VCMP tunnel initiation messages regarding new tunnel setup. The Gateway throttles incoming tunnel requests to the maximum rate they can be handled without disrupting the existing traffic, based on available cores. As a result, high queue length is expected in the queue on a Gateway with many tunnels.

The packet build up in these queues should come in large bursts following a specific event, like Gateway restart or transit interruption, and there should not be drops during normal operation.

vc_queue_vcmp_ctrl_0 and vc_queue_vcmp_ctrl_1 – This queue provides VCMP tunnel management control messages received on the existing tunnels. This includes messages such as route updates, path state updates, heartbeats, statistics, QoS Sync, and tunnel information.

Almost all control messages have built-in retry mechanisms to account for these drops, like route updates.

vc_queue_ike – The queue processes IKE protocol messages to manage keys and other state of encryption sessions.

This is generally a low volume traffic and it is unlikely that packet build up is encountered here. If drops occur, IKE messages are retried.

5.3.2 Monitor Throughput Performance

While handoff queues are the ideal way to monitor from a capacity perspective, it may be useful and/or interesting to monitor the throughput as well.

For many providers, the monitoring of throughput is done on the Hypervisor and is outside the scope of the Gateway.

For providers who want to monitor on the Gateway, the following example illustrates how to get the **RX** and **TX** byte counts, to make delta calculations over a period to measure the throughput.

Note: By default, DPDK is enabled.

```
vcadmin@vcg34-1:~$ sudo /opt/vc/bin/getcntr -c dpdk_eth0_pstat_ibytes -d vcgw.com 1895744358
 vcadmin@vcg34-1:~$ sudo /opt/vc/bin/getcntr -c dpdk_eth0_pstat_obytes -d vcgw.com 1865866321
 vcadmin@vcg34-1:~$ sudo /opt/vc/bin/getcntr -c dpdk_eth1_pstat_ibytes -d vcgw.com 33233362
 vcadmin@vcg34-1:~$ sudo /opt/vc/bin/getcntr -c dpdk_eth1_pstat_obytes -d vcgw.com 29843320
```

The actual throughput capacity might vary based on the number of connected Edges, encryption mix, and average packet size. The handoff queues provide a clear picture of the Gateway performance relative to its capacity.

For supported values, refer to the https://www.arista.com/en/support/product-documentation.

5.3.3 View Connected Edges

The following example shows the number of connected edges. It is recommended to have the tunnel count below the supported value to reduce the CPU load and recovery time that follows a restart.

```
vcadmin@vcg1-example:~$ /opt/vc/bin/debug.py --list_edges 2 { "vceCount": 156 } vcadmin@vcg1-example:~$
```

If the number of connected Edges approaches the supported value for a Gateway, then customers should be moved to alternate Gateways to reduce the Edge count. If the number of connected Edges exceeds the supported value for a Gateway, then this movement of Edges to alternate Gateways should be treated as critical.

Note: A single VeloCloud Gateway deployed with 4 cores supports a maximum of 2000 connected Edges. A Gateway deployed with 8 cores supports a maximum of 4000 connected Edges. You will need more Gateways in your pool for horizontal scale if you reach this limit.

5.3.4 Monitor Tunnel Count

You can monitor tunnel count with thresholds that provide warning or critical states which indicate potential issues prior to impacting services.

The following table lists the threshold values of tunnel count and recommended actions.

Table 4: Threshold State and Value for Tunnel Count

Threshold State	Threshold Value		Recommended Corrective Action
Warning/Critical	Gateway with 4 Cores and 32 GB of RAM		1. If tunnel count crosses warning or critical threshold: a. Collect diagnostic bundle. b. Check the stale tunnel count thresholds and perform corresponding actions listed for stale tunnels. c. If stale tunnels are within the warning threshold, it is recommended to quiesce the Gateway, add new Gateway to same Gateway pool and load balance the new Edge connections to another Gateway. d. If memory usage crosses critical threshold, restart services on Gateway. 2. If stale tunnels are observed beyond critical threshold: a. Collect core dump and diagnostic bundle. b. Restart the services on Gateway. c. Open a support case.
	With Certificate	Without Certificate	
	3000	3000	
	Gateway with 8 Cores and 32 GB of RAM		
	With Certificate	Without Certificate	
	6000	6000	

The following table lists the threshold values of stale tunnel count and recommended actions.

Table 5: Threshold State and Value for Stale Tunnel Count

Threshold State	Threshold Value	Recommended Corrective Action
Warning	10% of total tunnel count for a duration of 300 seconds	1. If stale tunnel count crosses warning or critical threshold, collect diagnostic bundle. 2. If stale tunnel count crosses critical threshold, open a support case.
Critical	25% of total tunnel count for a duration of 300 seconds	

5.3.5 Monitor Path Stability

You can monitor the status of unstable tunnel count to determine the path stability.

The following table lists the threshold values of unstable tunnel count and recommended actions.

Table 6: Threshold State and Value for Unstable Tunnel Count

Threshold State	Threshold Value	Recommended Corrective Action
Warning	25% of total tunnels in unstable state for 5 minutes	1. If unstable tunnel count crosses warning or critical threshold: • Collect diagnostic bundle. • Load balance new Edges to different Gateway. 2. If unstable tunnel count crosses critical threshold: • Open high priority support case along with diagnostic bundle.
Critical	25% of total tunnels in unstable state for 10 minutes	

5.3.6 View BGP-enabled VRFs

The following example shows the number of BGP-enabled VRFs.

```
vcadmin@vcg1-example:~$ /opt/vc/bin/debug.py --vrf | grep "my_asn" | wc -l 0 vcadmin@vcg1-
example:~$
```

If the number of BGP-enabled VRFs exceeds the maximum supported value, customers should be moved to alternate Gateways to reduce the number.

For supported values, refer to the https://www.arista.com/en/support/product-documentation.

5.3.7 View Gateway Routes

The following example shows the number of routes.

```
vcadmin@vcg1-example:~$ sudo /opt/vc/bin/getcntr -c memb.mod_gw_route_t.obj_cnt -d gwd-mem 8262
 vcadmin@vcg1-example:~$
```

If the number of route entries exceeds the maximum supported value, customers should be moved to alternate Gateways to reduce the route count.

For supported values, refer to the https://www.arista.com/en/support/product-documentation.

5.3.8 View Gateway Flows

The number of flows supported by a Gateway is determined by the system memory. There is a log that reflects the number of flows during startup.

The following example shows the log of maximum supported flows:

```
ERROR [MAIN] gwd_get_max_flow_supported:35 Flow Admission: GWD Max flow supported: 1929780 soft
 limit:1157820 hard limit:1736730
```

If logs have rolled over, use the following table as reference:

Table 7: Gateway Memory and Flows

Gateway Memory(GB)	Max Number of Flows	Critical Number of Flows(90% of max flows)
4	245760	221184
8	491520	442368
16	983040	884736
32	1966080	1769472

If flow limits reach a critical limit the system should be investigated for a possible flow leak.

Current flow objects in the system are as follows:

```
vcadmin@vcg1-example:~$ sudo /opt/vc/bin/getcntr -c memb.mod_mp_flow_t.obj_cnt -d gwd-mem
```

If the flows are determined to be invalid, a diagnostic bundle should be generated before restarting the Gateway service to clear the stale flows. If the flows are determined to be valid, then the customers should be moved to alternate Gateways to reduce the flow count.

The following table lists the threshold values and recommended actions for flow count.

Table 8: Threshold State and Values for Flow Count

Threshold State	Threshold Value	Recommended Corrective Action
Warning Critical	50% of 1.9 Million flows 75% of 1.9 Million flows	**1.** If total flow count crosses warning or critical threshold: • Collect diagnostic bundle. • Check the stale flow count thresholds and perform corresponding actions listed for stale flows. • If the stale flow count is within warning threshold, check the top consumers from flow table. • Disable any peer that is creating lot of rogue flows and if a DOS attack is suspected. • Check if any Enterprise is consuming most of the flow entries. This information can be used to load balance Edges in the Enterprise. • If memory usage crosses critical threshold, perform actions specified for memory metrics. **2.** If flow count crosses critical threshold: • Open high priority support case along with diagnostic bundle. • Restart the services on Gateway. • Use the following command to check the peers with high flow count: **`/opt/vc/bin/vc_top_peers.sh -t flow`**.

The following table lists the threshold values and recommended actions for stale flow count.

Table 9: Threshold State and Values for Stale Flow Count

Threshold State	Threshold Value	Recommended Corrective Action
Warning Critical	10% 25%	**1.** If stale flow count crosses warning or critical threshold: • Collect diagnostic bundle. • Check if small set of Edges are contributing to these stale flows. **2.** If stale flow count crosses critical threshold: • Open high priority support case, along with diagnostic bundle. • Restart the services on Gateway. **3.** If the same issue occurs multiple times on same Gateway or observed on different Gateways, mark the already created support case as **critical**.

5.3.9 View NAT Entries

If the number of free NAT entries is critically low, the system should be investigated for a possible leak.

```
vcadmin@vcg1-example:~$ sudo /opt/vc/bin/getcntr -c natd.nat_shmem_free_entries -d vcgwnat.com
 993408 vcadmin@vcg1-example:~$
```

Reboot the Gateway to clear all assigned NAT entries. Restarting the services has no effect on NAT entries.

For supported values, refer to the https://www.arista.com/en/support/product-documentation.

The following table lists the threshold values of NAT entries.

Table 10: Threshold States

Threshold State	Threshold Value	Recommended Corrective Action
Warning Critical	50% of 900K NAT entries 75% of 900K NAT entries	1. If total NAT count crosses warning or critical threshold: • Collect diagnostic bundle. • Check the stale NAT count thresholds and take corresponding actions listed for stale NAT count. • If the stale NAT count is within warning threshold, check the top consumers from NAT table. • Disable any peer that is creating lot of NAT entries, if a DOS attack is suspected. • Check if any Enterprise is consuming most of the NAT entries. This information can be used to load balance Edges in the Enterprise. • If all tenants are using NAT entries more or less equally and if memory usage crosses critical threshold, restart services on Gateway. 2. If NAT count crosses critical threshold: • Open high priority support case along with diagnostic bundle. • Restart the NAT services on Gateway and check if the issue is fixed. If not, restart all Gateway services. 3. Run the following command to check the peers with high NAT count: **`/opt/vc/bin/vc_top_peers.sh-t nat`**.

The following table lists the threshold values of stale AT entries.

Table 11: Threshold Values of Stale AT Entries

Threshold State	Threshold Value	Recommended Corrective Action
Warning Critical	10% 25%	**1.** If stale NAT count crosses warning or critical threshold: • Collect diagnostic bundle. • Check if small set of Edges are contributing to these stale NAT entries. **2.** If stale NAT count crosses critical threshold: • Open high priority support case along with diagnostic bundle and output of **`/opt/vc/bin/debug.py --stale_nat_dump`**. • Restart the NAT services on Gateway and check if the issue is fixed. If not, restart all Gateway services. **3.** If the same issue occurs multiple times on same Gateway or observed on different Gateways, mark the already created support case as **critical**.

5.3.10 Monitor Over Capacity Drops

Admission Control is a mechanism by which incoming data packets will be dropped when the system is at over capacity. This throttling helps in ensuring that the system has enough resources to process the already buffered packets. The admission control is applied only on data packets.

To check if there are any over capacity drops, use the following commands:

```
root@spperf-gateway-1:~# dispcnt -s over_capacity_drop over_capacity_drop = 1461980 0 /s
```

```
root@gateway-1:~# dispcnt -s over_capacity_drop -d vcgw.com Fri Dec 17 11:12:25 2021 over_capacity
_drop = 0 0 /s
```

```
root@gateway-1:~# dispcnt -s natd.shmem_oom -s natd.port_assign_fail -d vcgwnat.com Fri Dec 17
 11:12:44 2021 natd.port_assign_fail = 0 0 /s natd.shmem_oom = 0 0 /s
```

```
root@gateway-1:~# dispcnt -p netif -s tx_drop -s rx_drop -d vcgw.com Fri Dec 17 11:13:04 2021
 netif_eth0_rx_dropped = 0 0 /s netif_eth0_tx_dropped = 0 0 /s netif_eth1_rx_dropped = 0 0 /s
 netif_eth1_tx_dropped = 0 0 /s
```

To monitor the capacity of flows, run the following command:

```
root@gateway-1:~# dispcnt -s flow_admisison_limit_hit
```

To monitor the over capacity issues on NAT entries, run the following command:

```
root@gateway-1:~# dispcnt -s natd.shmem_oom -s natd.port_assign_fail -d vcgwnat.com Fri Dec 17
 11:12:44 2021 natd.port_assign_fail = 0 0 /s natd.shmem_oom = 0 0 /s
```

The following table lists the threshold values and recommended actions for overcapacity drops.

Table 12: Threshold Values for Overcapacity Drops

Threshold State	Threshold Value	Recommended Corrective Action

Warning	500 drops per 30 seconds (absolute count)
	When the drops remain above threshold value consistently for 5 minutes, warning alert is triggered.
		When the drops cross warning threshold:
		• Collect Gateway diagnostic bundle.
		• Check if a CPU intense system event is causing the packet drops.
		• Check flow metrics as follows:
		  • Maximum: 1.9M
		  • NAT: 960K
		  • Route: 1M for shared Gateway, 100K single enterprise
		• If throughput is consistently bursting for every 60 minutes or less to 2Gbps, quiesce the Gateway and add new Gateway to increase the capacity.
		• If any of the scale metrics have reached a 90% threshold of maximum limit, quiesce the Gateway and add new Gateway to increase the capacity.

Critical	1000 drops per 30 seconds (absolute count) When the drops remain above threshold value consistently for 5 minutes, critical alert is triggered.	When the drops cross critical threshold: • Collect Gateway diagnostic bundle. • Monitor throughput continuously for the next 15 minutes If the drops do not stabilize: • Quiesce the Gateway and add new Gateway to increase the capacity. • Check for top talker Enterprises to move to new Gateway. • At this stage Gateway is already causing user experience. After identifying the top talker Enterprises, rebalance the Edges immediately.

5.3.11 Monitor Latency Threshold for Paths

Whenever the latency threshold values are changed for an Edge, all the tunnels to the corresponding Gateway inherit the same threshold values. The debug command **`debug.py -v --path`** can be used to check the values.

Below is the sample output:

```
pi_info": { "connected": 2, "num_ha_takeover": 0, "priv_ip": "169.254.129.4", "profile":
 "0/0", "qoe_latency_threshold": { "trans_red_latency_ms": 100, "trans_yellow_latency_ms":
 100, "video_red_latency_ms": 50, "video_yellow_latency_ms": 10, "voice_red_latency_ms": 62,
 "voice_yellow_latency_ms": 22 }
```

The threshold values are not synced with other Edges or Hubs.

Chapter 6

Monitor Gateways using Telegraf

Telegraf is a plugin-driven server agent used for collecting and sending metrics and events from systems. You can configure Telegraf to collect the counters and statistics from an Input plugin and to send the data to an Output plugin.

To integrate Telegraf with Gateways to collect and export the counters to a third party plugin, see Configure Telegraf Integration.

6.1 Configure Telegraf Integration

Telegraf is a plugin-driven server agent for collecting and sending metrics and events from systems.

The Input plugin is **`/etc/telegraf/vcg_metrics.py`**, a file that contains the counters to be added. The Output plugin can be either **Wavefront** or **Prometheus**.

Telegraf collects the metrics from the declared Inputs and sends the details to the declared Outputs.

Note: Whenever there is a change in Telegraf configuration, you need to restart the Telegraf process using the command **`systemctl restart telegraf`**.

1. Use the following commands to configure the Output plugin. You can customize the ports in the corresponding configuration files as required.
   - **For Wavefront**

     ```
     [[outputs.wavefront]] host = " wavefront_proxy_IP" port = 2878 metric_separator = "."
      source_override = [" hostname", " agent_host", " node_host"] convert_paths = true
     ```

     The parameter *wavefront_proxy_IP* is the IP address of the Wavefront proxy server.

   - **For Prometheus**

     ```
     [[outputs.prometheus_client]] listen=":9273" metric_version=2
     ```

2. Telegraf needs to run the **`/opt/vc/bin/dispcnt`** command in **`vcg_metrics.sh`** to collect the metrics from Gateway, and the command requires sudo. Use the following command to add Telegraf to the sudo group.

   ```
   sudo usermod -G sudo telegraf
   ```

3. Add IP table rules to allow the external monitoring systems to get access to Telegraf port. The source IP address should be specified for security reasons. Add the following rules to allow traffic from wavefront and Prometheus. If required, you can customize the ports in the corresponding configuration files.

> **Note:** As IP table rules are not persistent across reboots, it is recommended to save the IP table rules using the command **`iptables-save`**. This command saves the rules automatically. You can also store the rules manually in a user-specific file and reuse the rules later.

- **For Wavefront**

```
sudo iptables -I INPUT -p tcp -m tcp --source <wavefront_proxy_IP>--sport 2878 -m comment --
comment "wavefront" -j ACCEPT
```

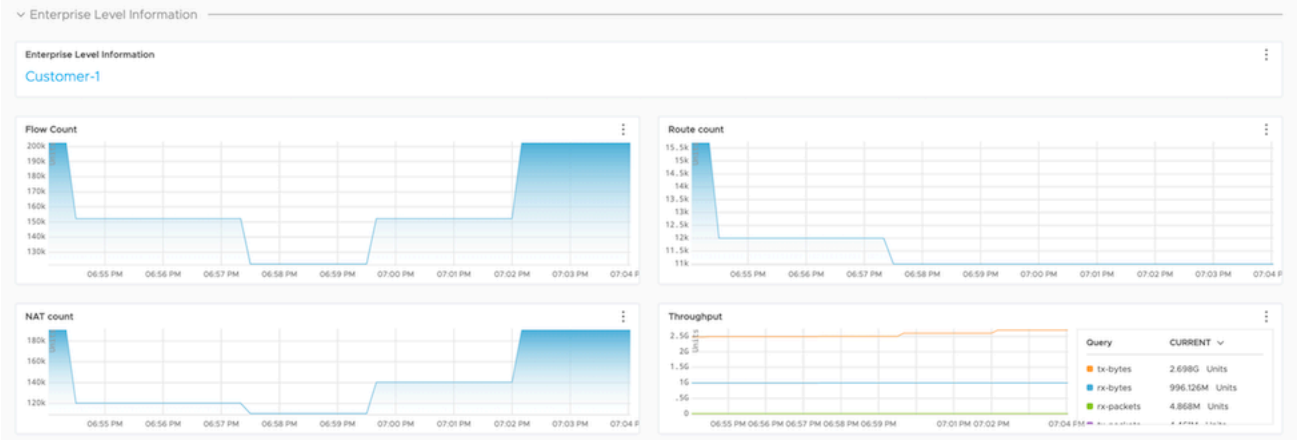
- **For Prometheus**

```
sudo iptables -I INPUT -p tcp -m tcp --source <IP>--dport 9273 -m comment --comment
 "prometheus" -j ACCEPT
```

The integration of Telegraf sends the data from the Gateways to the output plugins and you can view the details in the dashboards in visual format.

The following image shows an example output displayed in wavefront dashboard. The Graph illustrates Enterprise level information of flow count, NAT count, route count, and throughput details.

Figure 6-1: Example Output



For the list of the counters that are exported by the input plugin script **`/etc/telegraf/vcg_metrics.py`**, see Supported Counters.

6.2 Configure Telegraf as Syslog Receiver

You can configure Telegraf to receive Syslog settings from Gateways.

To configure Telegraf as syslog receiver:

1. In the **Operator** portal, select the **Gateway Management** tab and go to **Gateways** in the left navigation pane.
2. The **Gateways** page displays the list of available Gateways. Select the link to a Gateway. The details of the selected Gateway are displayed in the **Configure Gateways** page.

3. On the **Overview** tab, scroll down to the **Syslog Settings** section and configure Loopback IP address as the syslog receiver.

   Figure 6-2: Syslog Settings

4. Configure the Input plugin in the `/etc/telegraf/telegraf.conf` file with the protocol and port details configured in Orchestrator, using the following commands:

```
[[inputs.syslog]] server = "tcp://:6514" framing = "non-transparent"
```

   After configuring the Input plugin, make sure to restart the Telegraf service using the command **`systemctl restart telegraf`**.

5. Configure an Output plugin.

The integration of Telegraf sends the syslog data from the Gateways to the output plugins and you can view the details in the dashboards in visual format.

6.3 Supported Counters

After integrating Telegraf to an Gateway, you can collect the counters from the configured Input and export the data to the Output plugin.

The following table lists the supported Counters that can exported from an Gateway.

Table 13: Supported Counters from a Gateway

Counter Name	Description	Availability	Minimum Supported SD-WAN Version
number_of_edges	Number of Edges connected to the Gateway.	Global	4.3.0
number_of_tunnels	Number of tunnels associated with the Gateway.	Global	4.3.0
		IPv4, IPv6	4.5.0
number_of_routes	Number of routes installed in the Gateway.	Global	4.3.0
number_of_flows	Total number of active flows in the Gateway.	Global	4.3.0
active_NAT_entries	Number of active NAT entries per peer.	Global	4.3.0
free_NAT_entries	Number of free shared memory entries assigned for NAT.	Global	4.3.0
stale_NAT_entries	Number of stale NAT entries in the system. This tracks only the stale entries due to ref count leak.	Global	4.3.0
stale_tunnel_entries	Number of stale tunnel entries in the Gateway.	Global	4.3.0
stale_peer_objects	Number of stale peer objects in the Gateway.	Global	4.3.0
stale_flow_entries	Number of stale flow entries in the Gateway.	Global	4.3.0
sched_drop	Packets dropped in scheduler due to the bandwidth limit.	Global	4.3.0
flow_drop	Packets dropped due to flow lookup failure and flow creation failure.	Global	4.3.0
route_drop	Packets dropped due to route lookup failure and route sanity failure. Routing control packets which are dropped due to exceptions are also accounted.	Global	4.3.0
nat_drop	Packets dropped due to NAT lookup failure and NAT creation failure.	Global	4.3.0
over_capacity_drop	Packets dropped due to internal handoff queue limit drops and low packet buffers in the system.	Global	4.3.0
vcmp_drop	VCMP control and data packet dropped due to VCMP sanity checks and exceptions.	Global	4.3.0
invalid_pkt_drop	Packets dropped due to invalid checksum, TTL, and invalid packet size.	Global	4.3.0
misc_drop	Packets dropped due to other errors and exceptions.	Global	4.3.0
num_nsd_paths_up/down	Number of NSD tunnels in UP/ DOWN state in the Gateway.	Global	4.3.0

Counter Name	Description	Availability	Minimum Supported SD-WAN Version
num_paths_INITIAL	Number of tunnels in INITIAL state. INITIAL state indicates that the Edge just initiated a tunnel request to the Gateway.	Global IPv4, IPv6	4.3.0 4.5.0
num_paths _MEASURING_TX_BW num_paths _MEASURING_RX_BW	After initiating a tunnel request, tx and rx bandwidth will be measured for the tunnels from Edge to Gateway before moving to STABLE state. The number of tunnels for which tx and rx bandwidth is measured is tracked under the respective counters.	Global IPv4, IPv6	4.3.0 4.5.0
num_paths _STABLE	Number of tunnels in STABLE state. STABLE state indicates that the tunnel is established between Edge and Gateway and the tunnel is stable. To find the percentage of stable tunnels, multiply the number of stable tunnels by 100 and then divide that value by total number of tunnels.	Global IPv4, IPv6	4.3.0 4.5.0
num_paths _UNSTABLE	Number of tunnels in UNSTABLE state. If the loss, latency and jitter values exceed the defined threshold, the tunnel moves to UNSTABLE state. To find the percentage of unstable tunnels, multiply the number of unstable tunnels by 100 and then divide that value by total number of tunnels.	Global IPv4, IPv6	4.3.0 4.5.0
num_paths _QUIET	If no packets are received in the path for a defined time interval, path transitions to QUIET state and the number of such paths are tracked here.	Global IPv4, IPv6	4.3.0 4.5.0
nat_cnt	Number of active NAT entries per Enterprise.	Per Enterprise	4.3.0
route_cnt	Number of route entries installed in the Gateway per Enterprise.	Per Enterprise	4.3.0
flow_cnt	Total number of active flows per Enterprise.	Per Enterprise	4.3.0
tx_packets	Number of packets transmitted from the Gateway.	Per Enterprise, Per Edge Tunnel Per Non SD-WAN Destination	4.3.0 5.0.1 4.5.0

Counter Name	Description	Availability	Minimum Supported SD-WAN Version
tx_bytes	Number of bytes transmitted from the Gateway.	Per Enterprise	4.3.0
		Per Edge Tunnel	5.0.1
		Per Interface	5.0.1
		Per Non SD-WAN Destination	4.5.0
tx_errors	Number of packets dropped at TX due to packet errors.	Per Non SD-WAN Destination	5.1.0
rx_packets	Number of packets received by the Gateway.	Per Enterprise	4.3.0
		Per Non SD-WAN Destination	4.5.0
		Per Edge Tunnel	5.0.1
rx_bytes	Number of bytes received by the Gateway.	Per Enterprise	4.3.0
		Per Non SD-WAN Destination	4.5.0
		Per Edge Tunnel	5.0.1
		Per Interface	5.0.1
rx_errors	Number of packets dropped at RX due to packet errors.	Per Non SD-WAN Destination	5.1.0
vc_queue _<queue_name> _len	Number of packets enqueued in each handoff queues listed in Capacity of Gateway Components.	Global	4.3.0
vc_queue _<queue_name> _drop	Number of drops in each handoff queues listed in Capacity of Gateway Components.	Global	4.3.0
vc_queue _<queue_name> _wmark	Maximum number of packets enqueued in the respective queue at any point in time.	Global	4.5.1
vc_queue _<queue_name> _wmark_1min	Maximum number of packets enqueued in the respective queue in the last one minute.	Global	4.5.1
vc_queue _<queue_name> _wmark_5min	Maximum number of packets enqueued in the respective queue in the last 5 minutes.	Global	4.5.1
dpdk_mbuf_pending	Number of buffers that are already processed and waiting to be freed.	Global	4.3.0
dpdk_mbuf_locked_fail	Number of times the GET buffer operation fails while retrieving a buffer from the locked pool.	Global	4.3.0
dpdk_mbuf_locked_free	Number of free buffers present in the locked pool.	Global	4.3.0
dpdk_mbuf_pool_free	Number of free buffers.	Global	4.3.0
nombuf	Total number of RX mbuf allocation failures.	Global	4.3.0
mbuf_low	Number of times the mbuf_low threshold is reached.	Global	4.3.0
net_sch.pkt_cnt	Number of buffers used by Net Scheduler.	Global	4.3.0
link_sch.pkt_cnt	Number of buffers used by Link Scheduler.	Global	4.3.0
link_sch_cosq.pkt_cnt	Number of buffers used by Link Cos Scheduler.	Global	4.3.0

Counter Name	Description	Availability	Minimum Supported SD-WAN Version
mp.rt_pkts_stored	Number of buffers used by VCMP retransmit store.	Global	4.3.0
mp.reseq_qlen	Number of buffers used for VCMP resequencing.	Global	4.3.0
mp.jitter_pkt_bufs	Number of buffers used for VCMP jitter management.	Global	4.3.0
ipfrag.current_cnt	Number of buffers allocated for storing fragmented packets.	Global	4.3.0
crypto_drop	Number of packet drops observed due to crypto failures.	Global	4.3.0
frag_drop	Packet dropped due to fragmentation related issues.	Global	4.3.0
link_drop	Number of packet drops observed due to link specific issues.	Global	4.3.0
nat_over_capacity_drop	NAT over capacity drops due to port assignment failures.	Global	4.3.0
interface_over _capacity_drop	Packets dropped at the interface level due to over-capacity.	Global	4.3.0
misc_over_capacity_drop	Packets dropped due to internal handoff queue limit drops and due to low packet buffers in the system.	Global	4.3.0
<tx/rx>_pktsize _0_63	Number of packets with size 0-63 bytes transmitted/received in a Gateway.	Per Interface	4.5.0
<tx/rx>_pktsize _64_127	Number of packets with size 64-127 bytes transmitted/ received in a Gateway.	Per Interface	4.5.0
<tx/rx>_pktsize _128_255	Number of packets with size 128-255 bytes transmitted/ received in a Gateway.	Per Interface	4.5.0
<tx/rx>_pktsize _256_511	Number of packets with size 256-511 bytes transmitted/ received in a Gateway.	Per Interface	4.5.0
<tx/rx>_pktsize _512_1023	Number of packets with size 512-1023 bytes transmitted/ received in a Gateway.	Per Interface	4.5.0
<tx/rx>_pktsize _1024_1499	Number of packets with size 1024-1499 bytes transmitted/ received in a Gateway.	Per Interface	4.5.0
<tx/rx>_pktsize _1500	Number of packets with size above 1500 bytes transmitted/ received in a Gateway.	Per Interface	4.5.0
over_capacity_status	Indicates if a Gateway is running into over-capacity state due to internal handoff queue limit drops and low packet buffers in the system	Global	5.1.0
auto_rate_limit _drop	Packets dropped on Gateway due to auto rate-limit to restore Gateway from over capacity condition.	Global	5.2.0
red_factor	Rate limit factor set on VCMP peers based on RED applied.	Per Edge Tunnel	5.2.0

Counter Name	Description	Availability	Minimum Supported SD-WAN Version
capacity_metric _edge_count_value	Number of Edges connected to the Gateway.	Global	5.2.0
capacity_metric _edge_count _warning_threshold	Recommended threshold value (1500) for setting warning alerts based on the Edge count.	Global	5.2.0
capacity_metric _edge_count _critical_threshold	Recommended threshold value (2000) for setting critical alerts based on the Edge count.	Global	5.2.0
capacity_metric _tunnel_count _value	Number of tunnels associated with the Gateway.	Global	5.2.0
capacity_metric _tunnel_count _warning_threshold	Recommended threshold value (2625) for setting warning alerts based on the tunnel count.	Global	5.2.0
capacity_metric _tunnel_count _critical_threshold	Recommended threshold value (3500) for setting critical alerts based on the tunnel count.	Global	5.2.0
capacity_metric _pki_enabled _tunnel_count_value	Number of tunnels (with certificate) associated with the Gateway.	Global	5.2.0
capacity_metric_pki _enabled_tunnel_count _warning_threshold	Recommended threshold value (1875) for setting warning alerts based on the count of tunnels with certificate.	Global	5.2.0
capacity_metric_pki _enabled_tunnel_count _critical_threshold	Recommended threshold value (2250) for setting critical alerts based on the count of tunnels with certificate.	Global	5.2.0
capacity_metric _flow_count_value	Number of flows in the Gateway.	Global	5.2.0
capacity_metric_flow_count _warning_threshold	Recommended threshold value (475410) for setting warning alerts based on the flow count.	Global	5.2.0
capacity_metric_flow _count_critical_threshold	Recommended threshold value (713115) for setting critical alerts based on the flow count.	Global	5.2.0
capacity_metric _nat_count_value	Number of NAT entries in the Gateway.	Global	5.2.0
capacity_metric_nat _count_warning_threshold	Recommended threshold value (475410) for setting warning alerts based on the NAT count.	Global	5.2.0
capacity_metric_nat _count_critical_threshold	Recommended threshold value (713115) for setting critical alerts based on the NAT count.	Global	5.2.0
capacity_metric _pktq_wmark_value	Number of packet queue watermark in the Gateway.	Global	5.2.0
capacity_metric_pktq _wmark_warning _threshold	Recommended threshold value (2000) for setting warning alerts based on the packet queue watermark count.	Global	5.2.0
capacity_metric_pktq _wmark_critical_threshold	Recommended threshold value (6000) for setting critical alerts based on the packet queue watermark count.	Global	5.2.0
capacity_metric _pkt_drop_value	Number of packet drops in the Gateway.	Global	5.2.0

Counter Name	Description	Availability	Minimum Supported SD-WAN Version
capacity_metric_pkt_drop _warning_threshold	Recommended threshold value (500) for setting warning alerts based on the packet drops.	Global	5.2.0
capacity_metric_pkt_drop _critical_threshold	Recommended threshold value (2000) for setting critical alerts based on the packet drops.	Global	5.2.0
dpdk_xstats_<interface name>_tx_pps_<histogram bin counter range>	Histogram counters to monitor Gateway utilization based on egress packets per second. For packets per second (pps) monitoring, the defined histogram bin counter ranges are: • Up to 1000 pps • 1001 - 100000 pps • 100001 - 250000 pps • 250001 - 500000 pps • 500001 - 1000000 pps • More than 1000000 pps	Per DPDK Interface	6.0.0
dpdk_xstats_<interface name>_rx_pps_<histogram bin counter range>	Histogram counters to monitor Gateway utilization based on ingress packets per second.	Per DPDK Interface	6.0.0
dpdk_xstats_<interface name>_tx_mbps_<histogram bin counter range>	Histogram counters to monitor Gateway utilization based on egress throughput usage. For Megabits per second (Mbps) throughput monitoring, the defined histogram bin counter ranges are: • Up to 10 Mbps • 11 Mbps – 1000 Mbps • 1001 – 2000 Mbps • 2001 – 3000 Mbps • 3001 – 4000 Mbps • 4001 – 5000 Mbps • 5001 – 6000 Mbps • 6001 – 7000 Mbps • 7001 – 8000 Mbps • 8001 – 9000 Mbps • More than 9000 Mbps	Per DPDK Interface	6.0.0
dpdk_xstats_<interface name>_rx_mbps_<histogram bin counter range>	Histogram counters to monitor Gateway utilization based on ingress throughput usage.	Per DPDK Interface	6.0.0

Appendix A

References

A.1 Related Documents

The following documentation is available for ***Arista VeloCloud SD-WAN***:

- *Arista VeloCloud SD-WAN Operator Guide*
- *Arista VeloCloud SD-WAN Administration Guide*
- *Arista VeloCloud SD-WAN Partner Guide*
- *Arista VeloCloud SASE Global Settings Guide*
- *Arista VeloCloud SD-WAN Troubleshooting Guide*
- *Arista VeloCloud SD-WAN Orchestrator Deployment and Monitoring Guide*
- *Arista VeloCloud SD-WAN Gateway Monitoring Guide*
- *Arista VeloCloud SD-WAN API*
- *Arista VeloCloud Portal API*