Gateway Monitoring

VeloCloud SD-WAN 7.0

Version 7.0

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 Gateway Monitoring Guide

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

Version Compatibility

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

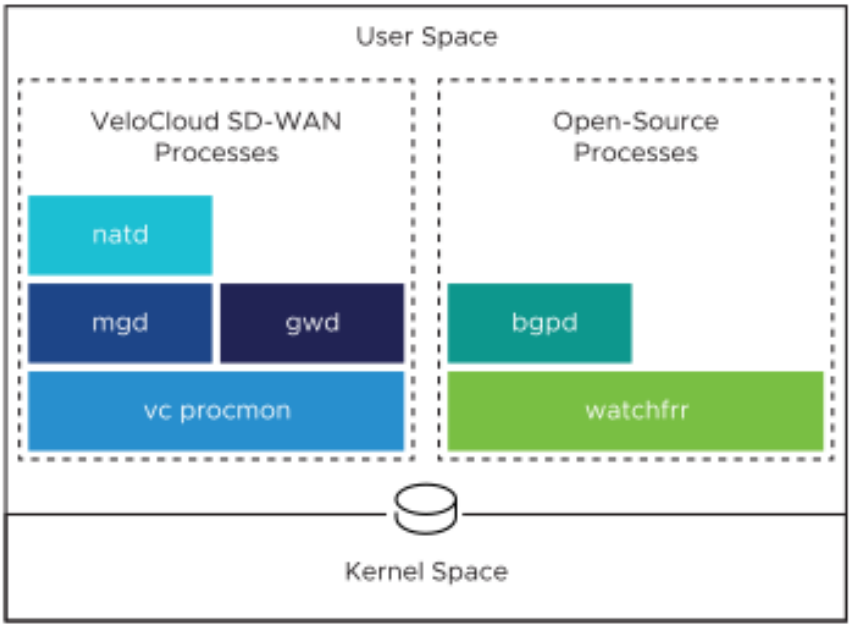
Chapter 2

Components

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

VeloCloud has implemented the entire networking stack in the user space in a process called `gwd`. While `gwd` contains the majority of functionality in the system, various other components 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`) provides the foundational process of the VeloCloud SD-WAN system. This process launches other VeloCloud SD-WAN processes, re-launches them on failure, and monitoring memory consumption of `gwd`.	`/var/log/vc_procmon.log`
`mgd`	The Management Plane Daemon (mgd) communicates with the Orchestrator. This process is kept isolated from gwd and if total failure of the gwd process occurs, the Orchestrator can obtain 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 such as 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 usage of the same NAT translations happens after the gwd restarts or upgrades. 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 responsible for launching **`bgpd`**, re-launching it on failure, and running any other related utilities. The script `/usr/sbin/frr.init` available on Gateway restarts 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 detects route failures between two connected entities faster with low-overhead detection of failures.	`/var/log/bfdd.log`

The Gateway uses the **gwd1** as an interface and provides the ability for **gwd** to deliver packets to the kernel. For example, 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 caches in the following file: `/opt/vc/.gateway.info`.

Use the script `/opt/vc/bin/getpolicy.py` to read the cached configuration and 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 is the Control Plane configuration. The Control Plane module contains sensitive information such as pre-shared keys for the Non SD-WAN Destination tunnels, and never caches locally. Therefore, if the Gateway restarts while disconnected to the Orchestrator, the components and processes that require a Control Plane blob such as Non SD-WAN Destinations, VLAN/VRF handoff, and BGP routing do not work until the Orchestrator connectivity restores.

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

Chapter 4

Monitor Gateways on VeloCloud Orchestrator

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

See Monitor Gateways.

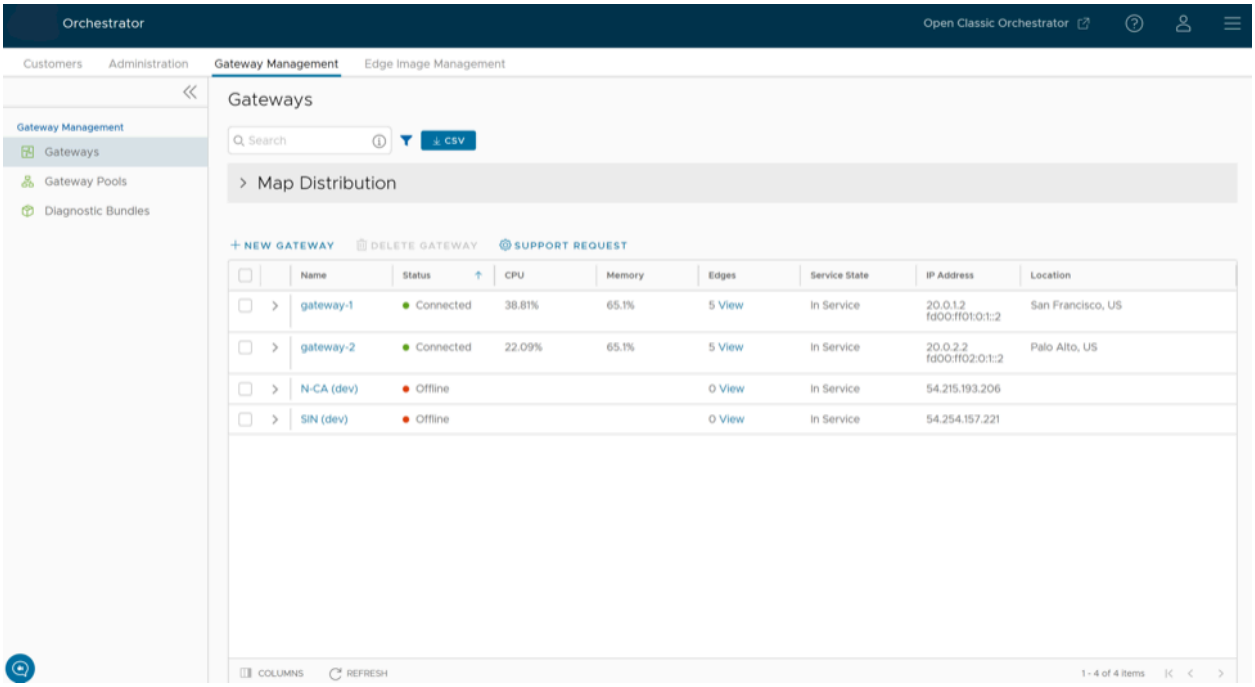
4.1 Monitor Gateways

Monitor the status and network usage data of Gateways available in the Operator and the Partner portal.

To monitor the Gateways, use the following steps:

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

Figure 4-1: Manage Gateways

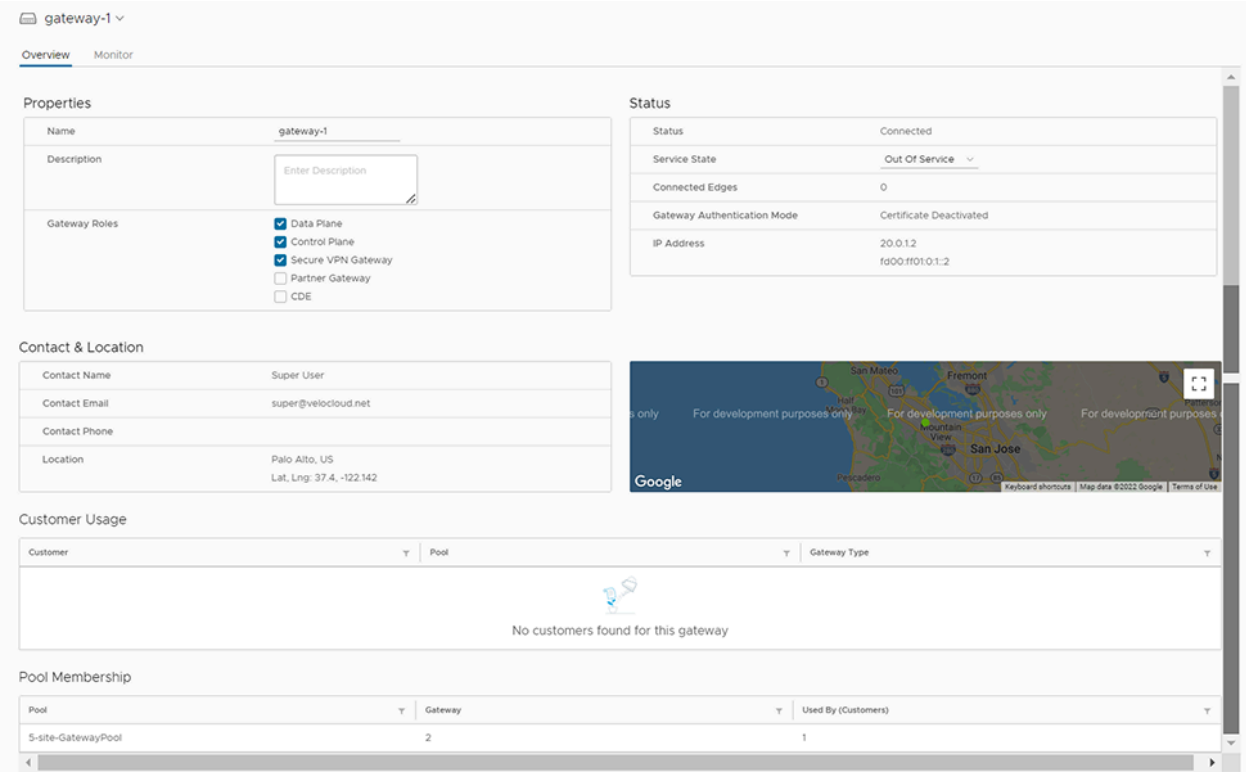


3. Select **Map Distribution** to expand and view the locations of the Gateways in the Map.
4. You can also select the arrows next to each Gateways name to view more details. The page displays the following details:
   - **Name** – Name of the Gateways.
   - **Status** – Current status of the Gateways. The status may be one of the following:
     - **Connected**

    - **Degraded**
    - **Never Activated**
    - **Not in Use**
    - **Offline**
    - **Out of Service**
    - **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**
    - **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.

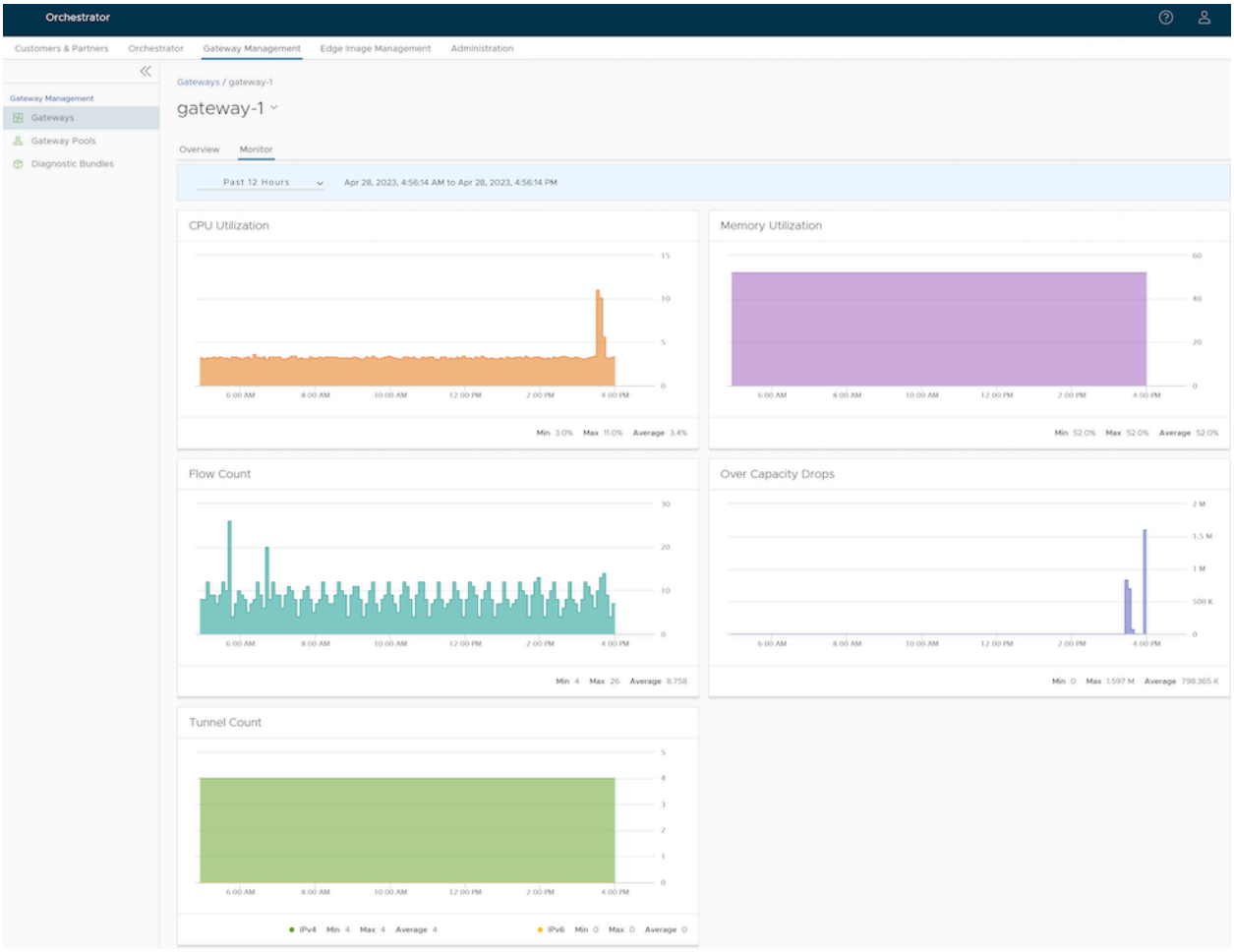
Figure 4-2: View Selected Gateway Details



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

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

Figure 4-3: View Usage Details of Selected Gateway



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. Expect occasional drops, 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 over 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

Use CLI commands to check the status of Gateways in an Orchestrator.

See the following topics:

- Monitor System Health
- Monitor VeloCloud SD-WAN Services
- Capacity of Gateway Components

5.1 Monitor System Health

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

This topic contains the following sections:

- Monitor Gateway Activation State
- View Activated Orchestrator Name
- View Software Version
- View NTP Time Zone
- View NTP Offset
- Monitor Disk Usage
- Monitor CPU Usage
- Monitor Memory Usage

5.1.1 Monitor Gateway Activation State

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

In the following example, the Gateway activated:

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

In the following example, the Gateway 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 locate the Orchestrator for the Gateway providing the Gateway activated.

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

5.1.3 View Software Version

The various VeloCloud processes in the system display version numbers which should all be identical.

Review 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:       0bcea3f6f0e6b8c21260187bb2d953e4cefd7f27
```

```
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:       0bcea3f6f0e6b8c21260187bb2d953e4cefd7f27
```

```
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 has an incorrect entry, 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 has set incorrectly, 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 such as 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 `/velocloud/vctcpdump.XXXX` with fixed size of 1MB. This file gets deleted when the session ends.

5.1.7 Monitor CPU Usage

The Gateway processes bursts of traffic and bursts of high CPU expected. The Gateway should be monitored for CPU cores 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 to indicate potential issues prior to impacting services. The following table lists the threshold values and recommended actions.

Table 2: Thresholds

Threshold State	Threshold Value		Recommended Corrective Action
	DP Core	Non DP Core	
Warning	95%	80%	If traffic crosses the threshold value consistently for 5 minutes: • Check per-process CPU usage. • Monitor for 10 more minutes. If the threshold value crosses consistently for 5 minutes: • Collect Gateway diagnostic bundle. • Open a support case with Arista.
Critical	98%	90%	If the threshold value crosses consistently for 5 minutes: • Monitor for possible critical packet drop which can indicate over capacity. If the issue observed for one hour: • If over capacity 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 more 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_support():
    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.isfile('/opt/vc/bin/debug.py'):
        f = subprocess.check_output(
            ["/opt/vc/bin/debug.py", "--dpdk_ports_dump"])
        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 `vc_process_monitor` monitors the memory of **gwd** and ensures that it never consumes more than 75% of available memory. As a result, monitoring for total system memory uses a warning threshold of 80% and critical threshold of 90%.

You can monitor a Gateway with thresholds to 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: Thresholds

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 observe again: • If over capacity 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 more 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 more 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["Buffers"] + 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.

This topic contains the following sections:

- Monitor VeloCloud SD-WAN Processes
- Monitor Certificate Revocation List
- Monitor ICMP Status
- Monitor BGP Sessions
- Monitor Core Files

5.2.1 Monitor VeloCloud SD-WAN Processes

The VeloCloud SD-WAN processes described in the Components should be running to ensure proper system functionality. Use the Linux command **`pgrep`** to identify the process. The format is slightly different for Python processes. When users execute the command, it returns the process ID (PID) if the process is currently active. If the process is inactive, the command returns an empty output.

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 remain in a Certificate Revocation List (CRL). If this list grows too long, generally due to an issue with the Certificate Authority of the Orchestrator, it impacts the performance of the Gateway. 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

When users enable the ICMP responder to track reachability for static routes on a Partner Gateway, the **`debug.py`** command displays the current operational state as **UP** or **DOWN**.

```
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:~$
```

Specifically, a DOWN state indicates that no Edges are currently connected to the Gateway, signaling that the path is inactive because there are no active tunnels to carry the traffic.

5.2.4 Monitor BGP Sessions

The **`bgp_view_summary debug.py`** provides information about the state of the Border Gateway Protocol (BGP) neighbor, and prefixes learned, and verifies that BGP is UP and exchanging prefixes.

To verify whether BGP neighborships have successfully formed, run the following command:

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

If the output shows that the BGP sessions are `down`, users should confirm that the Gateway maintains a stable connection to the Orchestrator.

5.2.5 Monitor Core Files

When a service crashes on the Gateway, the system automatically generates a core file that captures the process's state at the moment of failure. Users should retrieve the diagnostic bundles from the Orchestrator as soon as possible after this event occurs. After downloading the core file, users must share it along with the associated logs to Arista Support to facilitate a rapid root-cause analysis.

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

```
#! /usr/bin/env python
import subprocess
import traceback
import os
import os.path
import glob
import datetime
import time
import sys
import 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.isfile(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.isfile(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/crashlist.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=subprocess.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.Popen(
                            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.print_exc()
    helper.exit(summary="Nagios check could not complete",
                long_output=str(e), exit_code=unknown, 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(cmd)
        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 additional information on the capacity of different components and the supported values for each component, refer to the *VeloCloud SD-WAN Performance and Scale Datasheet* published at the **Partner**

Connect Portal. To access the datasheet, users must log in to the **Partner Connect Portal** using Partner credentials.

This topic contains the following sections:

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. Due to the bursty nature of traffic through a Gateway, expect occasional packet buildup in the packet forwarding queues. However, persistent high queue lengths in certain queues indicate a capacity problem.

The following example shows the output of the **`debug.py`** command for viewing the handoff queue output.

The output truncates to display only the first and last entries, for conciseness. Users 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:~$
```

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

The **qlength** column indicates the number of packets currently buffered in the queue. The **wmark** column indicates the maximum depth a queue has reached, indicating how close a Gateway came to dropping packets. The impact and remediation for these depend largely on the monitored queue.

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

This topic contains the following sections:

- Netif and Per-Core Queues
- View Non-Critical Queues

5.3.1.1 Netif and Per-Core Queues

The high watermark in these queues indicates that the packet processing rate is below 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 first column shows the current count, and the second column shows the rate per second. When users 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 calculated by dividing the lifetime total by 2. Since the data refreshes every two seconds, the first column displays the count since the last sample, and the second column displays 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_queue0_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_queue1_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 is less common or less likely to impact customers.

The following indicates the monitored non-critical queues.

vc_queue_vcmp_init - This queue provides VCMP tunnel-initiation messages for new tunnel set up. The Gateway throttles incoming tunnel requests to the maximum rate it can handle without disrupting existing traffic, based on available cores. Consequently, users should expect a high queue length on a Gateway that supports a large number of tunnels.

The packet buildup in these queues should occur in large bursts following a specific event, such as a Gateway restart or a transit interruption, and there should be no 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, Quality of Service (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 states of encryption sessions.

Given the low traffic volume, the system is unlikely to experience packet buildup at this point. If the Gateway drops a packet, the system automatically retries the IKE messages.

5.3.2 Monitor Throughput Performance

The system monitors handoff queues from a capacity perspective to prevent packet drops and congestion. Users may also find it useful to monitor throughput on these queues to gain a real-time view of data volume, helping to identify peak usage patterns and forecast when users might need to scale your Gateway resources.

For many providers, throughput monitoring occurs on the Hypervisor, outside the Gateway's scope.

For providers monitoring the Gateway, the following example illustrates how to obtain the **RX** and **TX** byte counts and perform delta calculations over a period to measure throughput.

Note: By default, the system enables the Data Plane Development Kit (DPDK).

```
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's performance relative to its capacity.

For the supported maximum throughput value, refer to the *VeloCloud SD-WAN Performance and Scale Datasheet* published on the **Partner Connect Portal**. To access the datasheet, users must log in to the **Partner Connect Portal** using Partner credentials (username and password).

5.3.3 View Connected Edges

The following example shows the number of connected edges. Arista recommends keeping the tunnel count below the supported limit to reduce CPU load and the recovery time after 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 limit for a Gateway, customers should migrate to alternative Gateways to reduce the Edge count. When the number of connected Edges exceeds the Gateway’s supported capacity, users must treat the migration of Edges to alternate Gateways as a critical priority.

Note: A single VeloCloud Gateway with four cores supports up to 2000 connected Edges. A Gateway with eight cores supports up to 4000 connected Edges. You need more Gateways in your pool for horizontal scale if you reach this limit.

5.3.4 Monitor Tunnel Count

Users can monitor tunnel count with thresholds to trigger warnings or critical states that indicate potential issues before they impact services.

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

Table 4: Threshold Values

Threshold State	Threshold Value		Recommended Corrective Action
Warning/Critical	Gateway with 4 Cores and 32 GB of RAM		1. If the tunnel count crosses the 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, VeloCloud recommends quiescing the Gateway, adding a new Gateway to the same Gateway pool, and load-balancing new Edge connections to the new Gateway. d. If memory usage exceeds the critical threshold, restart services on the Gateway. 2. If stale tunnels are beyond the critical threshold: a. Collect core dump and diagnostic bundle. b. Restart the services on the Gateway. c. Open a support case with Arista.
	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 stale tunnel count threshold values and recommended actions.

Table 5: Threshold Values

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

5.3.5 Monitor Path Stability

Users can monitor the unstable tunnel count to determine path stability.

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

Table 6: Threshold States

Threshold State	Threshold Value	Recommended Corrective Action
Warning	25% of the total tunnels are in an unstable state for 5 minutes	1. If unstable tunnel count crosses the warning or critical threshold: • Collect diagnostic bundle. • Load balance the new Edges to different Gateways. 2. If unstable tunnel count crosses critical threshold: • Open a high-priority support case with Arista and include the diagnostic bundle.
Critical	25% of the total tunnels are in an 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:~$
```

When the number of BGP-enabled VRFs exceeds the maximum supported value, ensure customers are moved to alternate Gateways to reduce load.

For supported values, refer to the *VeloCloud SD-WAN Performance and Scale Datasheet* published at the **Partner Connect Portal**. To access the datasheet, users must log in to the **Partner Connect Portal** using Partner credentials (username and password).

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, ensure customers are moved to alternate Gateways to reduce the route count.

For supported values, refer to the *VeloCloud SD-WAN Performance and Scale Datasheet* published at the **Partner Connect Portal**. To access the datasheet, users must log in to the **Partner Connect Portal** using Partner credentials (username and password).

5.3.8 View Gateway Flows

System memory directly determines the maximum number of flows a Gateway can support. A log file reflects the number of flows during start-up.

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 a reference:

Table 7: Maximum Supported 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 counts reach critical limits, users must investigate the system for a potential 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 a user determines the flows are invalid, the user must ensure to generate the diagnostic bundle before restarting the Gateway service to flush the stale entries. However, if the user confirms the flows are valid, ensure customers are moved to alternate Gateways to redistribute the load and reduce the flow count on the oversubscribed unit.

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

Table 8: Threshold States

Threshold State	Threshold Value	Recommended Corrective Action
Warning	50% of 1.9 Million flows	**1.** If the total flow count crosses the warning or critical threshold: • Collect diagnostic bundle. • Verify the stale flow count thresholds and perform the corresponding actions listed for stale flows. • If the stale flow count is within the warning threshold, check the top consumers from flow table. • Disable any peer that is creating a lot of rogue flows, and if a DoS attack is suspected. • Verify whether a specific Enterprise is consuming an excessive number of flow entries. By analyzing this data, users can load balance the Edges within that Enterprise across multiple Gateways. • If memory usage exceeds the critical threshold, perform the actions specified for the memory metrics. **2.** If the flow count crosses the critical threshold: • Open a high-priority support case with Arista, along with a diagnostic bundle. • Restart the services on the Gateway. • Run the following command to check the peers with high flow count: `/opt/vc/bin/vc_top_peers.sh -t flow`.
Critical	75% of 1.9 Million flows	

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

Table 9: Threshold States

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

5.3.9 View NAT Entries

If the number of free Network Address Translation (NAT) entries is critically low, the system investigates 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 does not affect NAT entries.

For supported values, refer to the *VeloCloud SD-WAN Performance and Scale Datasheet* published at the Partner Connect Portal. To access the datasheet, users must log in to the Partner Connect Portal using Partner credentials (username and password).

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 the total NAT count crosses the warning or critical threshold: • Collect diagnostic bundle. • Verify the stale NAT count thresholds and take the corresponding actions listed for each threshold. • If the stale NAT count is within the warning threshold, check the top consumers from the NAT table. • Disable any peer that is creating many NAT entries if a DOS attack is suspected. • Verify if all tenants are using NAT entries more or less equally, and if memory usage crosses the critical threshold, restart services on the Gateway. **2.** If the NAT count crosses the critical threshold: • Open a high-priority support case with Arista, along with a diagnostic bundle. • Restart the NAT services on the Gateway and verify 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 the warning or critical threshold: • Collect diagnostic bundle. • Verify if a small set of Edges is contributing to these stale NAT entries. **2.** If stale NAT count crosses critical threshold: • Open high-priority support case with Arista, along with diagnostic bundle and output of /opt/vc/bin/debug.py--stale_nat_dump. • Restart the NAT services on Gateway and verify if the issue is fixed. If not, restart all Gateway services. **3.** If the same issue occurs multiple times on the same Gateway or across different Gateways, mark the existing support case as **critical**.

5.3.10 Monitor Overcapacity Drops

Admission Control is a mechanism that drops incoming data packets when the system is over capacity. This throttling helps ensure the system has enough resources to process the already-buffered packets. The admission control is applied only to data packets.

To check if there are any overcapacity drops, run 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 overcapacity issues on Network Address Translation (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 the threshold value consistently for 5 minutes, a warning alert is triggered.
		When the drops cross the warning threshold:
		• Collect the 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 90% of the maximum limit, quiesce the Gateway and add a new Gateway to increase capacity.

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

5.3.11 Monitor Latency Threshold for Paths

When users update the latency threshold values for an Edge, all tunnels connecting to the corresponding Gateway automatically inherit those new settings. Users can verify these active thresholds by running the command **`debug.py-v--path`**.

The following 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 system does not sync these threshold values with other Edges or Hubs in the network.

Chapter 6

Monitor Gateways using Telegraf

Telegraf is a plugin-driven server agent that collects and sends metrics and events from systems. Users 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.

Also, see the following topics:

- Configure Telegraf as Syslog Receiver
- Supported Counters

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 the Telegraf configuration changes, you need to restart the Telegraf process with 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 = truewavefront_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 the 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 the 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<wavefront_proxy_IP>--sport 2878 -m comment --comment "wavefront"
 -j ACCEPT
```

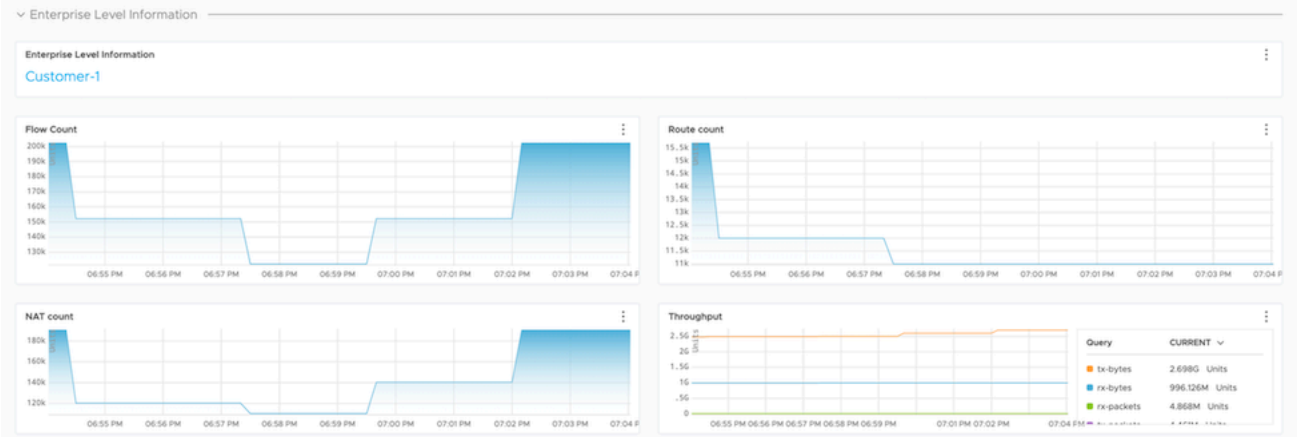
For Prometheus

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

The following image shows an example output displayed in the 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 a 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 display on the **Configure Gateways** page.
3. On the **Overview** tab, scroll down to the **Syslog Settings** section and configure the 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 a visual format.

6.3 Supported Counters

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

The following table lists the supported counters exported from a Gateway.

Table 13: Supported Counters

Counter Name	Description	Availability	Minimum Supported SD-WAN Version
active_NAT_entries	Number of active NAT entries per peer.	Global	4.3.0
auto_rate_limit_drop	Packets dropped on the Gateway due to the auto rate-limit to restore the Gateway from over capacity condition.	Global	5.2.0
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 (2250) for setting warning alerts based on the Edge count.	Global	5.2.0
capacity_metric_edge_count_critical_threshold	Recommended threshold value (2375) 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 (3600) for setting warning alerts based on the tunnel count.	Global	5.2.0
capacity_metric_tunnel_count_critical_threshold	Recommended threshold value (3800) 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 (3600) for setting warning alerts based on the count of tunnels with a certificate.	Global	5.2.0
capacity_metric_pki_enabled_tunnel_count_critical_threshold	Recommended threshold value (3800) for setting critical alerts based on the count of tunnels with a 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
capacity_metric_edge_count_congestion_threshold	Recommended threshold value for setting congestion alerts based on the Edge count.	Global	5.2.0
capacity_metric_tunnel_count_congestion_threshold	Recommended threshold value for setting congestion alerts based on the Tunnel count.	Global	5.2.0
capacity_metric_pki_enabled_tunnel_count_congestion_threshold	Recommended threshold value for setting congestion alerts based on the count of tunnels with certificates.	Global	5.2.0
capacity_metric_flow_count_congestion_threshold	Recommended threshold value for setting congestion alerts based on the Flow count.	Global	5.2.0
capacity_metric_nat_count_congestion_threshold	Recommended threshold value for setting congestion alerts based on the NAT count.	Global	5.2.0
capacity_metric_pktq_wmark_congestion_threshold	Recommended threshold value for setting congestion alerts based on the packet queue watermark count.	Global	5.2.0
capacity_metric_pkt_drop_congestion_threshold	Recommended threshold value for setting congestion alerts based on the packet drop.	Global	5.2.0
crypto_drop	Number of packet drops observed due to crypto failures.	Global	4.3.0
dce_info_check_new_cnt	Number of times a Hub's DCE info was not found for an Enterprise, indicating no Hub endpoint information exists yet, and a fresh DCE info population is required.	Per Enterprise	4.3.0
dce_info_check_zero_cnt	Number of times a Hub's DCE info was found but contained zero endpoint entries, indicating the assigned Hub has stale or empty endpoint information and requires repopulation.	Per Enterprise	4.3.0
dce_info_check_nz_cnt	Number of times a Hub's DCE info was found with valid non-zero endpoint entries, indicating the assigned Hub has active endpoint information, and no repopulation is needed.	Per Enterprise	4.3.0
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

Counter Name	Description	Availability	Minimum Supported SD-WAN Version
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
flow_count	Total number of active flows per Enterprise.	Per Enterprise	4.3.0
flow_drop	Packets dropped due to flow lookup failure and flow creation failure.	Global	4.3.0
flow_over_capacity_drop	Packets dropped due to the Gateway's flow table reaching its maximum capacity and being unable to allocate new flows.	Global	4.3.0

Counter Name	Description	Availability	Minimum Supported SD-WAN Version
frag_drop	Packet dropped due to fragmentation related issues.	Global	4.3.0
free_NAT_entries	Number of free shared memory entries assigned for NAT.	Global	4.3.0
invalid_pkt_drop	Packets dropped due to invalid checksum, TTL, and invalid packet size.	Global	4.3.0
interface_over_capacity_drop	Packets dropped at the interface level due to over-capacity.	Global	4.3.0
ipfrag_current_cnt	Number of buffers allocated for storing fragmented packets.	Global	4.3.0
link_drop	Number of packet drops observed due to link specific issues.	Global	4.3.0
link_sch_cosq_pkt_cnt	Number of buffers used by Link Cos Scheduler.	Global	4.3.0
link_sch_pkt_cnt	Number of buffers used by Link Scheduler.	Global	4.3.0
misc_drop	Packets dropped due to other errors and exceptions.	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
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
nat_cnt	Number of active NAT entries per Enterprise.	Per Enterprise	4.3.0
nat_drop	Packets dropped due to NAT lookup failure and NAT creation failure.	Global	4.3.0
nat_over_capacity_drop	NAT over capacity drops due to port assignment failures.	Global	4.3.0
net_sch_pkt_cnt	Number of buffers used by Net Scheduler.	Global	4.3.0
netif_<interface>_pktsize_0_63	Number of packets with size 0-63 bytes transmitted/received in a Gateway.	Per Interface	4.5.0
netif_<interface>_pktsize_64_127	Number of packets with size 64-127 bytes transmitted/received in a Gateway.	Per Interface	4.5.0
netif_<interface>_pktsize_128_255	Number of packets with size 128-255 bytes transmitted/received in a Gateway.	Per Interface	4.5.0
netif_<interface>_pktsize_256_511	Number of packets with size 256-511 bytes transmitted/received in a Gateway.	Per Interface	4.5.0
netif_<interface>_pktsize_512_1023	Number of packets with size 512-1023 bytes transmitted/received in a Gateway.	Per Interface	4.5.0
netif_<interface>_pktsize_1024_1499	Number of packets with size 1024-1499 bytes transmitted/received in a Gateway.	Per Interface	4.5.0
netif_<interface>_pktsize_1500	Number of packets with size above 1500 bytes transmitted/received in a Gateway.	Per Interface	4.5.0

Counter Name	Description	Availability	Minimum Supported SD-WAN Version
netif_queue<core>_len	The current number of packets in the network interface that receive a queue for the specified dataplane core.	Global	5.0.0
netif_queue<core>_wmark	Maximum number of packets enqueued in the network interface that receive a queue for the specified dataplane core at any point in time.	Global	6.2.0
netif_queue<core>_wmark_1s	The maximum number of packets enqueued in the network interface that receive a queue for the specified dataplane core in the last second.	Global	6.2.0
netif_queue<core>_wmark_1min	The maximum number of packets enqueued in the network interface that receive a queue for the specified dataplane core in the last one minute.	Global	6.2.0
netif_queue<core>_wmark_5min	The maximum number of packets enqueued in the network interface that receive a queue for the specified dataplane core in the last 5 minutes.	Global	6.2.0
nombuf	Total number of RX mbuf allocation failures.	Global	4.3.0
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
number_of_tunnels_v4 number_of_tunnels_v6	Number of IPv4/IPv6 tunnels associated with the Gateway	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
num_nsd_paths_up/down	Number of NSD tunnels in UP/DOWN state in the Gateway.	Global	4.3.0
num_paths_INITIAL	Number of tunnels in INITIAL state.	Global	4.3.0
num_pathsv4_INITIAL num_pathsv6_INITIAL	INITIAL state indicates that the Edge just initiated a tunnel request to the Gateway.	IPv4, IPv6	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	4.3.0
num_pathsv4_MEASURING_TX_BW num_pathsv4_MEASURING_RX_BW		IPv4, IPv6	4.5.0
num_pathsv6_MEASURING_TX_BW num_pathsv6_MEASURING_RX_BW			

Counter Name	Description	Availability	Minimum Supported SD-WAN Version
num_paths_PATH_STABLE	Number of tunnels in STABLE state.	Global	4.3.0
num_pathsv4_STABLE num_pathsv6_STABLE	STABLE state indicates that the tunnel is established between Edge and Gateway and is stable. To find the percentage of stable tunnels, multiply the number of stable tunnels by 100 and then divide that value by the total number of tunnels.	IPv4, IPv6	4.5.0
num_paths_UNSTABLE	Number of tunnels in UNSTABLE state.	Global	4.3.0
num_pathsv4_UNSTABLE num_pathsv6_UNSTABLE	If the loss, latency, and jitter values exceed the defined threshold, the tunnel moves to the UNSTABLE state. To find the percentage of unstable tunnels, multiply the number of unstable tunnels by 100 and then divide that value by the total number of tunnels.	IPv4, IPv6	4.5.0
num_paths_QUIET num_pathsv4_QUIET num_pathsv6_QUIET	If no packets are received in the path for a defined time interval, the path transitions to the QUIET state, and the number of such paths is tracked here.	Global IPv4, IPv6	4.3.0 4.5.0
num_paths_active	Number of Tunnels in the ACTIVE state.	Global	4.3.0
num_pathsv4_active	Number of Tunnels in the ACTIVE state.	IPv4	4.5.0
num_pathsv6_active	Number of Tunnels in the ACTIVE state.	IPv6	4.5.0
num_pathsv4_BW_UNMEASURABLE num_pathsv6_BW_UNMEASURABLE	Number of VCMP tunnel paths where bandwidth measurement has been attempted but failed. Paths in this state will automatically retry bandwidth measurement after 30 minutes.	IPv4, IPv6	4.5.0
num_pathsv4_WAITING_FOR_LINK_BW num_pathsv6_WAITING_FOR_LINK_BW	Number of VCMP tunnel paths that have completed initial set up but are waiting for another path on the same link to finish its bandwidth measurement first. The path remains in this state until the link's TX and RX bandwidth values are populated by another path's measurement, at which point it inherits those values and transitions toward a stable state.	IPv4, IPv6	4.5.0
over_capacity_drop	Packets dropped due to internal handoff queue limit drops and low packet buffers in the system.	Global	4.3.0
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
per_core_queue<core>_len	The current number of packets in the per-core processing queue for the specified dataplane core.	Global	5.0.0
per_core_queue<core>_wmark	The maximum number of packets enqueued in the per-core processing queue for the specified dataplane core at any point in time.	Global	6.2.0

Counter Name	Description	Availability	Minimum Supported SD-WAN Version
per_core_queue<core>_wmark_1s	The maximum number of packets enqueued in the per-core processing queue for the specified dataplane core in the last one second.	Global	6.2.0
per_core_queue<core>_wmark_1min	The maximum number of packets enqueued in the per-core processing queue for the specified dataplane core in the last one minute.	Global	6.2.0
per_core_queue<core>_wmark_5min	The maximum number of packets enqueued in the per-core processing queue for the specified dataplane core in the last 5 minutes.	Global	6.2.0
red_factor	Rate limit factor set on VCMP peers based on RED applied.	Per Edge Tunnel	5.2.0
route_count	Number of route entries installed in the Gateway per Enterprise.	Per Enterprise	4.3.0
route_drop	Packets dropped due to route lookup failure and route sanity failure. Routing control packets that are dropped due to exceptions are also accounted.	Global	4.3.0
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_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
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 by the scheduler due to bandwidth limits.	Global	4.3.0
tx_bytes	Number of bytes transmitted from the Gateway.	Per Enterprise	4.3.0

Counter Name	Description	Availability	Minimum Supported SD-WAN Version
		Per Edge Tunnel	5.0.1
		Per Interface	5.0.1
		Per Non SD-WAN Destination	4.5.0
tx_packets	Number of packets transmitted from the Gateway.	Per Enterprise,	4.3.0
		Per Edge Tunnel	5.0.1
		Per Non SD-WAN Destination	4.5.0
vcmp_drop	VCMP control and data packet dropped due to VCMP sanity checks and exceptions.	Global	4.3.0
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>_drops	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_1s	Maximum number of packets enqueued in the respective queue in the last one second.	Global	6.2.0
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

Note: The threshold values vary by Gateway. The values in the table are for a 4-CPU Gateway.

Appendix A

References

This topic contains the following sections:

A.1 Related Documents

Arista provides the following documentation for **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 Design Guide for Enhanced Firewall Services*
- *Arista VeloCloud SD-WAN 6.4 API*
- *Arista VeloCloud Portal API 6.4*