New Features Guide

SD-WAN with FortiOS, FortiManager, and FortiAnalyzer 8.0.x

FORTINET DOCUMENT LIBRARY
https://docs.fortinet.com

FORTINET VIDEO LIBRARY
https://video.fortinet.com

FORTINET BLOG
https://blog.fortinet.com

CUSTOMER SERVICE & SUPPORT
https://support.fortinet.com

FORTINET TRAINING & CERTIFICATION PROGRAM
https://www.fortinet.com/training-certification

FORTINET TRAINING INSTITUTE
https://training.fortinet.com

FORTIGUARD LABS
https://www.fortiguard.com

END USER LICENSE AGREEMENT
https://www.fortinet.com/doc/legal/EULA.pdf

FEEDBACK
Email: techdoc@fortinet.com

April 28, 2026
SD-WAN with FortiOS, FortiManager, and FortiAnalyzer 8.0.x New Features Guide
01-80X-1276494-20260428

TABLE OF CONTENTS

Change Log

Date	Change Description
2026-04-28	Initial release for FortiOS and FortiManager 8.0.0. See What's new in 8.0.0 on page 5.

Overview

This guide provides details of new features for SD-WAN introduced in FortiOS 8.0, FortiManager 8.0, and FortiAnalyzer 8.0.

- What's new in 8.0.0 on page 5

For each feature, the guide provides detailed information on configuration, requirements, and limitations, as applicable. For features introduced in FortiManager or FortiAnalyzer, the short product name is appended to the end of the topic heading, for example: Interface-based bandwidth graph uses average bandwidth logic FMG on page 54.

What's new in 8.0.0

Feature	Details
Application performance	• ToS matching and negate options on adaptive FEC profiles on page 6 • Stop packet duplication upon reaching bandwidth utilization threshold on page 13 • Use secondary tunnel for FEC redundant parity packets on page 19 • ToS-based SD-WAN duplication matching on page 26 • On-demand duplication at the Hub using remote health-check on page 31 • Confine packet duplication across multiple overlays to the same spoke on page 40 • SD-WAN speed test scheduling improvements on page 45
Operations	• Interface-based bandwidth graph uses average bandwidth logic FMG on page 54
SD-WAN steering	• SD-WAN underlay bandwidth steers traffic on page 56 • Selective SD-WAN duplication on page 63 • Control SD-WAN interface usage based on monthly traffic volume (quota) on page 68

Application performance

8.0.0

ToS matching and negate options on adaptive FEC profiles

This information is also available in the FortiOS 8.0 Administration Guide:

- ToS matching and negate options on adaptive FEC profiles

Adaptive FEC now has the ability to take the ToS/DSCP value into consideration when mapping the FEC profile. In the first round of matching, traffic will be matched against packet-loss, latency, and bandwidth. In the second round of matching, the TOS value and mask will be used. In addition, new negate options allow users to match against the negative of a threshold. For example, negate packet-loss of 5% means any packet-loss less than 5% will be matched.

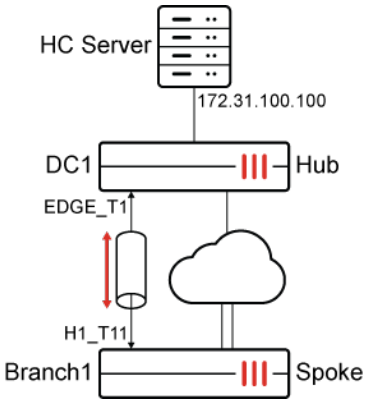
For more information about how to configure ToS value and netmask and how it maps to DSCP values, please see Configuring SD-WAN rules.

New CLI:

```
config vpn ipsec fec
    edit <name>
        config mappings
            edit <id>
                set packet-loss-threshold-negate {enable | disable}
                set latency-threshold-negate {enable | disable}
                set bandwidth-up-threshold-negate {enable | disable}
                set bandwidth-down-threshold-negate {enable | disable}
                set bandwidth-bi-threshold-negate {enable | disable}
                config tos
                    edit <id>
                        set tos <value>
                        set tos-mask <value>
```

```
                        set base <value>
                        set redundant <value>
                    next
                end
            next
        end
    next
end
```

Example



In this single-hub SD-WAN deployment, the spoke has defined a performance health-check targeting the HC Server behind the Hub. Adaptive FEC is configured based on the health-checks using the following criteria:

	Health Check	Matching tos 0x01 mask 0xff	Matching tos 0x02 mask 0xff	Not matching either traffic class
Rule 1	Packet-loss is above the threshold of 5%	Apply base 5 redundant 1	Apply base 5 redundant 2	Apply default base 11 redundant 1
Rule 2	Latency is less than the threshold of 20ms	Apply base 6 redundant 1	Apply base 6 redundant 2	Apply default base 12 redundant 2
Rule 3	Available upstream bandwidth is more than 50Mbps	Apply base 7 redundant 1	Apply base 7 redundant 2	Apply default base 13 redundant 3
Rule 4	Packet-loss is below the threshold of 5%	Do not send any redundant packets		

The rules are examined in the order that they are configured.

To configure the spoke:

1. Configure SD-WAN:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
        edit "overlay"
        next
        edit "underlay"
        next
    end
    config members
        ...
        edit 3
            set interface "H1_T11"
            set zone "overlay"
            set source 172.31.0.65
        next
        ...
    end
    config health-check
        edit "HUB"
            set server "172.31.100.100"
            set update-static-route disable
            set members 3
            config sla
                edit 1
                next
            end
        next
    end
    config service
        edit 1
            set dst "CORP_LAN"
            set priority-members 3
        next
    end
end
```

2. Configure VPN:

```
config vpn ipsec fec
    edit "fec_mapping"
        config mappings
            edit 1
                set base 11
                set redundant 1
                set packet-loss-threshold 5
                config tos
                    edit 1
```

```
                    set tos 0x01
                    set tos-mask 0xff
                    set base 5
                    set redundant 1
                next
                edit 2
                    set tos 0x02
                    set tos-mask 0xff
                    set base 5
                    set redundant 2
                next
            end
        next
        edit 2
            set base 12
            set redundant 2
            set latency-threshold 20
            config tos
                edit 1
                    set tos 0x01
                    set tos-mask 0xff
                    set base 6
                    set redundant 1
                next
                edit 2
                    set tos 0x02
                    set tos-mask 0xff
                    set base 6
                    set redundant 2
                next
            end
        next
        edit 3
            set base 13
            set redundant 3
            set bandwidth-up-threshold 50000
            config tos
                edit 1
                    set tos 0x01
                    set tos-mask 0xff
                    set base 7
                    set redundant 1
                next
                edit 2
                    set tos 0x02
                    set tos-mask 0xff
                    set base 7
                    set redundant 2
                next
            end
        next
```

```
            edit 4
                set base 40
                set redundant 0
                set packet-loss-threshold 5
                set packet-loss-threshold-negate enable
            next
        end
    next
end
```

```
config vpn ipsec phase1-interface
    edit "H1_T11"
        ...
        set fec-egress enable
        set fec-codec rs
        set fec-health-check "HUB"
        set fec-mapping-profile "fec_mapping"
        ...
    next
end
```

3. Configure a firewall policy:

```
config firewall policy
    edit 1
       ...
        set fec enable
    next
end
```

To configure the hub:

```
config vpn ipsec phase1-interface
    edit "EDGE_T1"
        ...
        set fec-ingress enable
        ...
    next
end
```

To test the configuration:

1. Check SD-WAN metrics on H1_T11 and the applied FEC settings:
   In this first scenario, the first 3 rules were not matched. Therefore, the 4th rule implies that no redundant packets are sent.

```
Branch1_A_FGT (root) (Interim)# diagnose sys sdwan health-check
Health Check(HUB):
Seq(3 H1_T11): state(alive), packet-loss(0.000%), latency(30.247), jitter(0.016), mos(4.389),
custom_profile(0.000), bandwidth-up(49758), bandwidth-dw(89990), bandwidth-bi(139748), sla_
map=0x0
```

```
Branch1_A_FGT# diagnose vpn  tunnel  fec  H1_T11
egress:  enabled=1
        base=40 redundant=0 codec=0 timeout=10(ms)
        encode=0 encode_timeout=0 encode_fail=0
        tx_data=0 tx_parity=0
        total_tx_data=0 total_tx_parity=0
ingress: enabled=0
        timeout=0(ms) fasm_cnt=0 fasm_full=0
        ipsec_fec_chk_fail=0 complete=0
        recover=0 recover_timeout=0 recover_fail=0
        rx_data=0 rx_parity=0
        rx=0 rx_fail=0
```

2. Make the available upstream bandwidth more than 50M by decreasing traffic on H1_T11, then check the applied FEC settings. The third rule is matched:

```
# diagnose sys sdwan health-check
Health Check(HUB):
Seq(3 H1_T11): state(alive), packet-loss(0.000%), latency(30.234), jitter(0.022), mos(4.389),
custom_profile(0.000), bandwidth-up(63244), bandwidth-dw(89993), bandwidth-bi(153237), sla_
map=0x0

Branch1_A_FGT (root) (Interim)# diagnose vpn  tunnel  fec  H1_T11
egress:  enabled=1
        base=13 redundant=3 codec=0 timeout=10(ms)
        encode=0 encode_timeout=0 encode_fail=0
        tx_data=0 tx_parity=0
        tos/mask=0x01/0xFF:
                base=7 redundant=1
                encode=0 encode_timeout=0 encode_fail=0
                tx_data=0 tx_parity=0
        tos/mask=0x02/0xFF:
                base=7 redundant=2
                encode=0 encode_timeout=0 encode_fail=0
                tx_data=0 tx_parity=0
        total_tx_data=0 total_tx_parity=0
ingress: enabled=0
        timeout=0(ms) fasm_cnt=0 fasm_full=0
        ipsec_fec_chk_fail=0 complete=0
        recover=0 recover_timeout=0 recover_fail=0
        rx_data=0 rx_parity=0
        rx=0 rx_fail=0
```

3. Make latency on H1_T11 less than 20ms, then check the applied FEC settings again. Both the latency and upstream bandwidth are above the threshold, so because rule 2 comes first, rule 2 is applied:

```
# diagnose sys sdwan  health-check
Health Check(HUB):
Seq(3 H1_T11): state(alive), packet-loss(0.000%), latency(0.247), jitter(0.027), mos(4.404),
custom_profile(0.000), bandwidth-up(89999), bandwidth-dw(89997), bandwidth-bi(179996), sla_
map=0x1
```

```
Branch1_A_FGT (root) (Interim)# diagnose vpn  tunnel  fec  H1_T11
egress:  enabled=1
        base=12 redundant=2 codec=0 timeout=10(ms)
        encode=0 encode_timeout=0 encode_fail=0
        tx_data=0 tx_parity=0
        tos/mask=0x01/0xFF:
                base=6 redundant=1
                encode=0 encode_timeout=0 encode_fail=0
                tx_data=0 tx_parity=0
        tos/mask=0x02/0xFF:
                base=6 redundant=2
                encode=0 encode_timeout=0 encode_fail=0
                tx_data=0 tx_parity=0
        total_tx_data=0 total_tx_parity=0
ingress: enabled=0
        timeout=0(ms) fasm_cnt=0 fasm_full=0
        ipsec_fec_chk_fail=0 complete=0
        recover=0 recover_timeout=0 recover_fail=0
        rx_data=0 rx_parity=0
        rx=0 rx_fail=0
```

4. Make packet-loss on H1_T11 more than 5%, then check the applied FEC settings again. All 3 rules match since health checks are above the threshold, so because rule 1 comes first, rule 1 is applied.

```
# diagnose sys sdwan  health-check
Health Check(HUB):
Seq(3 H1_T11): state(alive), packet-loss(6.000%), latency(0.254), jitter(0.029), mos(4.401),
custom_profile(0.000), bandwidth-up(89999), bandwidth-dw(89997), bandwidth-bi(179996), sla_
map=0x0

Branch1_A_FGT (root) (Interim)# diagnose vpn  tunnel  fec  H1_T11
egress:  enabled=1
        base=11 redundant=1 codec=0 timeout=10(ms)
        encode=0 encode_timeout=0 encode_fail=0
        tx_data=0 tx_parity=0
        tos/mask=0x01/0xFF:
                base=5 redundant=1
                encode=0 encode_timeout=0 encode_fail=0
                tx_data=0 tx_parity=0
        tos/mask=0x02/0xFF:
                base=5 redundant=2
                encode=0 encode_timeout=0 encode_fail=0
                tx_data=0 tx_parity=0
        total_tx_data=0 total_tx_parity=0
ingress: enabled=0
        timeout=0(ms) fasm_cnt=0 fasm_full=0
        ipsec_fec_chk_fail=0 complete=0
        recover=0 recover_timeout=0 recover_fail=0
        rx_data=0 rx_parity=0
        rx=0 rx_fail=0
```

Stop packet duplication upon reaching bandwidth utilization threshold

Utilized bandwidth on overlays or their underlays can be checked before duplication. When utilized bandwidth is less than the specified threshold, duplication starts. When it exceeds the specified threshold, duplication stops.

New options are available to specify the bandwidth measurement to use for evaluation:

```
config system sdwan
    config members
        edit <id>
            set duplication-threshold-upbandwidth <integer>
            set duplication-threshold-dwbandwidth <integer>
            set duplication-threshold-bibandwidth <integer>
        next
    end
end
```

Option	Description
`duplication-threshold-upbandwidth <integer>`	Bandwidth upstream threshold value in kilobytes per second (0 - 4294967295, default = 0).
`duplication-threshold-dwbandwidth <integer>`	Bandwidth downstream threshold value in kilobytes per second (0 - 4294967295, default = 0).
`duplication-threshold-bibandwidth <integer>`	Bandwidth bistream threshold value in kilobytes per second (0 - 4294967295, default = 0).

A new option is available to specify underlay or overlay:

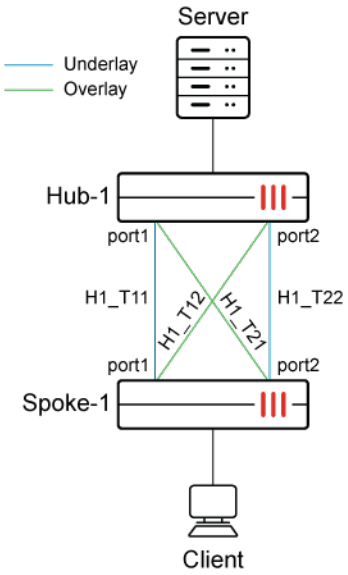
```
config system sdwan
    config members
        edit <id>
            set duplication-threshold-bandwidth {overlay* | underlay}
        next
    end
end
```

Option	Description
`duplication-threshold-bandwidth {overlay* | underlay}`	Configure duplication threshold bandwidth interface in the SD-WAN. Choose from: • `overlay`: Use overlay interface bandwidth for duplication in the SD-WAN. • `underlay`: Use underlay interface bandwidth for duplication in the SD-WAN.

Example

This example demonstrates how to configure and test a threshold for bandwidth utilization. In summary:

- Spoke-1 has four overlays: H1_T11, H1_T12, H1_T21, and H1_T22.
- H1_T11 and H1_T12 are associated with the same underlay port1, and H1_T21 and H1_T22 are associated with the same underlay port2.
- Traffic is intended to duplicate on all four overlays as long as the used bandwidth on the underlays doesn't exceed the specified threshold.
- Once the used bandwidth exceeds the specified threshold, duplication on the associated overlays stops.



To configure Spoke-1:

1. Configure Spoke-1.

   In this example, the duplication maximum is set to 4. For 4 members, the upstream bandwidth is set to 50 M, and the used bandwidth of its underlay is referenced. Duplication is also configured.

```
config system sdwan
    set status enable
    set duplication-max-num 4
    config zone
        edit "virtual-wan-link"
        next
        edit "overlay"
        next
        edit "underlay"
        next
    end
    config members
        edit 1
            set interface "port1"
            set zone "underlay"
```

```
        next
        edit 2
            set interface "port2"
            set zone "underlay"
            set gateway 172.31.3.6
        next
        edit 3
            set duplication-threshold-upbandwidth 50000
            set duplication-threshold-bandwidth underlay
            set interface "H1_T11"
            set zone "overlay"
            set source 172.31.0.65
        next
        edit 4
            set duplication-threshold-upbandwidth 50000
            set duplication-threshold-bandwidth underlay
            set interface "H1_T12"
            set zone "overlay"
            set source 172.31.0.65
        next
        edit 5
            set duplication-threshold-upbandwidth 50000
            set duplication-threshold-bandwidth underlay
            set interface "H1_T21"
            set zone "overlay"
            set source 172.31.0.65
        next
        edit 6
            set duplication-threshold-upbandwidth 50000
            set duplication-threshold-bandwidth underlay
            set interface "H1_T22"
            set zone "overlay"
            set source 172.31.0.65
        next
    end
     config health-check
        edit "HUB"
            set server "172.31.100.100"
            set update-static-route disable
            set members 3 4 5 6
            config sla
                edit 1
                next
            end
        next
    end
    config duplication
        edit 1
            set srcaddr "CORP_LAN"
            set dstaddr "CORP_LAN"
            set srcintf "lan_zone"
```

```
            set dstintf "overlay"
            set service "PING"
            set packet-duplication force
        next
    end
end
```

2. Send constant pings from the client to server to duplicate traffic on all overlays.
   The following example shows duplication on four overlays:

```
test@PC3:~$ ping 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=62 time=0.593 ms
64 bytes from 10.0.1.2: icmp_seq=1 ttl=62 time=0.628 ms (DUP!)
64 bytes from 10.0.1.2: icmp_seq=1 ttl=62 time=0.636 ms (DUP!)
64 bytes from 10.0.1.2: icmp_seq=1 ttl=62 time=0.643 ms (DUP!)
64 bytes from 10.0.1.2: icmp_seq=2 ttl=62 time=0.579 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=62 time=0.592 ms (DUP!)
64 bytes from 10.0.1.2: icmp_seq=2 ttl=62 time=0.594 ms (DUP!)
64 bytes from 10.0.1.2: icmp_seq=2 ttl=62 time=0.595 ms (DUP!)
^C
--- 10.0.1.2 ping statistics ---
2 packets transmitted, 2 received, +6 duplicates, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.579/0.607/0.643/0.033 ms

# diagnose sniffer packet any 'host 10.0.1.2' 4
interfaces=[any]
filters=[host 10.0.1.2]
0.700706 port4 in 10.0.3.2 -> 10.0.1.2: icmp: echo request
0.700740 H1_T12 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
0.700761 H1_T21 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
0.700782 H1_T22 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
0.700796 H1_T11 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
0.701139 H1_T11 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
0.701164 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
0.701170 H1_T11 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
0.701182 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
0.701188 H1_T11 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
0.701203 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
0.701208 H1_T11 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
0.701220 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
1.700691 port4 in 10.0.3.2 -> 10.0.1.2: icmp: echo request
1.700730 H1_T12 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
1.700759 H1_T21 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
1.700777 H1_T22 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
1.700792 H1_T11 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
1.701115 H1_T11 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
1.701144 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
1.701150 H1_T11 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
1.701163 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
1.701168 H1_T11 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
1.701180 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
```

```
1.701185 H1_T11 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
1.701196 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
```

3. Generate around 80 M background traffic on underlay port2.

   Duplication on H1_T21 and H1_T22, which are associated with port2, stops because 80 M background traffic on underlay port2 exceeds the threshold 50 M. Duplication occurs on H1_T11 and H1_T12:

```
# diagnose sys sdwan intf-sla-log port2

Timestamp: Tue Sep 30 09:42:18 2025, ifn: port2, used inbandwidth: 5363bps, used outbandwidth:
84142940bps, used bibandwidth: 84148303bps, tx bytes: 95718653738bytes, rx bytes:
304404953bytes.
Timestamp: Tue Sep 30 09:42:28 2025, ifn: port2, used inbandwidth: 5152bps, used outbandwidth:
84143025bps, used bibandwidth: 84148177bps, tx bytes: 95823831406bytes, rx bytes:
304410822bytes.

test@PC3:~$ ping 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=62 time=0.734 ms
64 bytes from 10.0.1.2: icmp_seq=1 ttl=62 time=0.744 ms (DUP!)
64 bytes from 10.0.1.2: icmp_seq=2 ttl=62 time=0.508 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=62 time=0.519 ms (DUP!)
^C
--- 10.0.1.2 ping statistics ---
2 packets transmitted, 2 received, +2 duplicates, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.508/0.626/0.744/0.114 ms
test@PC3:~$

# diagnose sniffer packet any 'host 10.0.1.2 and icmp' 4
interfaces=[any]
filters=[host 10.0.1.2 and icmp]
2.631011 port4 in 10.0.3.2 -> 10.0.1.2: icmp: echo request
2.631117 H1_T12 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
2.631139 H1_T11 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
2.631587 H1_T11 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
2.631618 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
2.631624 H1_T11 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
2.631637 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
3.629973 port4 in 10.0.3.2 -> 10.0.1.2: icmp: echo request
3.630001 H1_T12 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
3.630023 H1_T11 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
3.630361 H1_T11 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
3.630387 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
3.630393 H1_T11 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
3.630406 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
```

4. Stop 80 M background traffic on port2.

   Duplication on four overlays occurs again:

```
# diagnose sys sdwan intf-sla-log port2

Timestamp: Tue Sep 30 09:54:30 2025, ifn: port2, used inbandwidth: 7239bps, used outbandwidth:
```

```
7964bps, used bibandwidth: 15203bps, tx bytes: 101144616164bytes, rx bytes: 304801192bytes.
Timestamp: Tue Sep 30 09:54:40 2025, ifn: port2, used inbandwidth: 7898bps, used outbandwidth:
7957bps, used bibandwidth: 15855bps, tx bytes: 101144625934bytes, rx bytes: 304812991bytes.

test@PC3:~$ ping 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=62 time=0.663 ms
64 bytes from 10.0.1.2: icmp_seq=1 ttl=62 time=0.712 ms (DUP!)
64 bytes from 10.0.1.2: icmp_seq=1 ttl=62 time=0.719 ms (DUP!)
64 bytes from 10.0.1.2: icmp_seq=1 ttl=62 time=0.722 ms (DUP!)
64 bytes from 10.0.1.2: icmp_seq=2 ttl=62 time=0.561 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=62 time=0.611 ms (DUP!)
64 bytes from 10.0.1.2: icmp_seq=2 ttl=62 time=0.613 ms (DUP!)
64 bytes from 10.0.1.2: icmp_seq=2 ttl=62 time=0.614 ms (DUP!)
^C
--- 10.0.1.2 ping statistics ---
2 packets transmitted, 2 received, +6 duplicates, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.561/0.651/0.722/0.066 ms

# diagnose sniffer packet any 'host 10.0.1.2 and icmp' 4
interfaces=[any]
filters=[host 10.0.1.2 and icmp]
3.019665 port4 in 10.0.3.2 -> 10.0.1.2: icmp: echo request
3.019778 H1_T12 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
3.019800 H1_T21 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
3.019819 H1_T22 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
3.019833 H1_T11 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
3.020200 H1_T22 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
3.020240 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
3.020247 H1_T22 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
3.020260 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
3.020266 H1_T22 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
3.020277 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
3.020282 H1_T22 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
3.020293 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
4.018668 port4 in 10.0.3.2 -> 10.0.1.2: icmp: echo request
4.018722 H1_T12 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
4.018744 H1_T21 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
4.018759 H1_T22 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
4.018772 H1_T11 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
4.019100 H1_T22 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
4.019133 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
4.019139 H1_T22 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
4.019152 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
4.019157 H1_T22 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
4.019170 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
4.019174 H1_T22 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
4.019186 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
^C
26 packets received by filter
0 packets dropped by kernel
```

Use secondary tunnel for FEC redundant parity packets

This information is also available in the FortiOS 8.0 Administration Guide:

- Use secondary tunnel for FEC redundant parity packets

In the original implementation of adaptive FEC, FEC parity packets are sent inside the same overlay tunnel. However, the parity packets are susceptible to being dropped at the same rate as other VPN packets.

This enhancement to adaptive FEC introduces the capability to send FEC parity packets in a secondary overlay tunnel that is associated with the same destination. The following setting can be configured on the primary tunnel that requires parity packets to be sent on a redundant tunnel:

```
config vpn ipsec phase1-interface
    edit <tunnel>
        set fec-separate-redundant-tunnel enable
    next
end
```

The redundant tunnel must have the same destination `location-id` as the original tunnel.

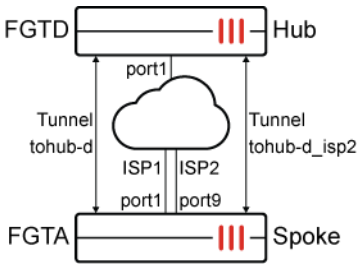
Example

In the following example, a VPN Hub (FGTD) is configured as a dialup VPN gateway. A spoke has two underlays going to different ISPs, where the primary (port1) has 10G bandwidth and the secondary (port9) has 1G bandwidth. Two tunnels are established from the Spoke to the Hub over the two underlays. The intention is to send data over the primary tunnel while using the secondary tunnel for redundancy and parity packets for the primary tunnel.

The FEC mappings are configured to send two parity packets per 8 base packets upon packet-loss threshold above 2%. Additionally, it will send three parity packets per nine base packets when bandwidth exceeds 950Mbps.

This example utilizes one Hub/destination, so the default location-id will be shared between the tunnels without any conflicts. When multiple Hubs are in use, and you need to apply the parity packet duplication on the same Hub and destination, use a different location-id for each hub.

```
config system settings
    set location-id <unique id for each hub>
end
```



To configure the hub, FGTD:

```
config vpn ipsec phase1-interface
    edit "tospokes"
        set type dynamic
        set interface "port1"
        set ike-version 2
        set peertype any
        set net-device disable
        set mode-cfg enable
        set proposal aes128-sha256 aes256-sha256 aes128gcm-prfsha256 aes256gcm-prfsha384
chacha20poly1305-prfsha256
        set add-route disable
        set dpd on-idle
        set dhgrp 20 21
        set auto-discovery-sender enable
        set fec-ingress enable
        set transport auto
        set ipv4-start-ip 100.100.100.10
        set ipv4-end-ip 100.100.100.20
        set ipv4-netmask 255.255.255.0
        set psksecret ****************
        set dpd-retryinterval 60
    next
end
```

```
config vpn ipsec phase2-interface
    edit "tospokes"
        set phase1name "tospokes"
        set proposal aes128-sha256 aes256-sha256 aes128gcm aes256gcm chacha20poly1305
        set dhgrp 20 21
    next
end
```

To configure the spoke, FGTA:

1. Configure FEC mapping to bind network SLA metrics and FEC base and redundant packets:

```
config vpn ipsec fec
    edit "m1"
        config mappings
            edit 1
                set base 8
                set redundant 2
                set packet-loss-threshold 2
            next
            edit 2
                set base 9
                set redundant 3
                set bandwidth-up-threshold 950000
            next
```

```
        end
    next
end
```

2. Configure the IPsec tunnels:

   On the primary tunnel, enable FEC enabled and apply FEC mapping. The redundant tunnel does not need FEC enabled, or the fec-separate-redundant-tunnel setting.

```
config vpn ipsec phase1-interface
    edit "tohub-d"
        set interface "port1"
        set ike-version 2
        set peertype any
        set net-device disable
        set mode-cfg enable
        set proposal aes128-sha256 aes256-sha256 aes128gcm-prfsha256 aes256gcm-prfsha384
chacha20poly1305-prfsha256
        set add-route disable
        set dhgrp 20 21
        set auto-discovery-receiver enable
        set auto-discovery-shortcuts dependent
        set fec-egress enable
        set fec-separate-redundant-tunnel enable
        set fec-codec rs
        set fec-health-check "1"
        set fec-mapping-profile "m1"
        set transport auto
        set remote-gw 172.16.200.4
        set psksecret ****************
    next
    edit "tohub-d_isp2"
        set interface "port9"
        set ike-version 2
        set peertype any
        set net-device disable
        set mode-cfg enable
        set proposal aes128-sha256 aes256-sha256 aes128gcm-prfsha256 aes256gcm-prfsha384
chacha20poly1305-prfsha256
        set add-route disable
        set dhgrp 20 21
        set auto-discovery-receiver enable
        set auto-discovery-shortcuts dependent
        set transport auto
        set remote-gw 172.16.200.4
        set psksecret ****************
    next
end
```

```
config vpn ipsec phase2-interface
    edit "tohub-d"
        set phase1name "tohub-d"
```

```
        set proposal aes128-sha256 aes256-sha256 aes128gcm aes256gcm chacha20poly1305
        set dhgrp 20 21
        set auto-negotiate enable
    next
    edit "tohub-d_isp2"
        set phase1name "tohub-d_isp2"
        set proposal aes128-sha256 aes256-sha256 aes128gcm aes256gcm chacha20poly1305
        set dhgrp 20 21
        set auto-negotiate enable
    next
end
```

3. Configure SD-WAN:

   Assign the overlay interface as an SD-WAN member and attach it to a zone. Configure health-check on the overlay member in order to detect packet-loss and bandwidth utilization.

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "port1"
            set gateway 172.16.200.3
        next
        edit 4
            set interface "tohub-d"
        next
        edit 5
            set interface "port9"
            set gateway 11.101.1.2
        next
    end
    config health-check
            edit "1"
            set server "192.168.5.44"
            set members 4
            config sla
                edit 1
                next
            end
        next
    end
    config service
        edit 2
            set name "1_clone"
            set dst "all"
            set src "10.1.100.0"
            set priority-members 4 1
        next
```

```
    end
end
```

To verify the results:

1. Confirm that both tunnels are established on their respective interfaces on FGTA:

```
# diagnose vpn ike gateway list

vd: root/0
name: tohub-d_isp2
version: 2
interface: port9 15
addr: 11.101.1.1:5929 -> 172.16.200.4:4500
ext addr: 11.101.1.1 5929
tun_id: 10.0.0.2/::10.0.0.2
remote_location: 0.0.0.0
network-id: 0
transport: TCP
virtual-interface-addr: remote: 100.100.100.4
created: 407s ago
peer-id: 172.16.200.4
peer-id-auth: no
assigned IPv4 address: 100.100.100.11/255.255.255.0
nat: me
auto-discovery: 2 receiver
pending-queue: 0
PPK: no
IKE SA: created 1/1  established 1/1  time 36000/36000/36000 ms
IPsec SA: created 1/1  established 1/1  time 36000/36000/36000 ms

  id/spi: 10217 77f9b317d1c985cd/cd40650eb175ec62
  direction: initiator
  status: established 407-371s ago = 36000ms
  proposal: aes128-sha256
  child: no
  SK_ei: 3fc9a5151af50e58-fec5cfed3c436fb1
  SK_er: de07bdc42f053f82-cdc945fc5a7ad5f4
  SK_ai: 0262f60ee41bafa2-2f7b9fc40f38e2b6-415b31885b81b778-b28c62a9915d2e5d
  SK_ar: 693d5e98d6695490-bc9c26b8a15e5322-0285e96df10e48cc-3535038e2c9a442d
  PPK: no
  message-id sent/recv: 2/0
  QKD: no
  PQC-KEM (IKE): no
  PQC-KEM (all IPsec): no
  lifetime/rekey: 86400/85728
  DPD sent/recv: 00000000/00000000
  peer-id: 172.16.200.4

vd: root/0
name: tohub-d
version: 2
```

```
interface: port1 7
addr: 172.16.200.1:500 -> 172.16.200.4:500
ext addr: 172.16.200.1 500
tun_id: 172.16.200.4/::172.16.200.4
remote_location: 0.0.0.0
network-id: 0
transport: UDP
virtual-interface-addr: remote: 100.100.100.4
created: 377s ago
peer-id: 172.16.200.4
peer-id-auth: no
assigned IPv4 address: 100.100.100.10/255.255.255.0
auto-discovery: 2 receiver
pending-queue: 0
PPK: no
IKE SA: created 1/1  established 1/1  time 0/0/0 ms
IPsec SA: created 1/1  established 1/1  time 0/0/0 ms

  id/spi: 10218 43bc6decd5bc56af/d5a405f1ff957a2f
  direction: initiator
  status: established 377-377s ago = 0ms
  proposal: aes128-sha256
  child: no
  SK_ei: 677aadd417ae719f-6cc99b2215be8711
  SK_er: f2b69ab1aecc08d6-04236a6849b56cae
  SK_ai: e9294b9b9087f973-9aebb458b27f5b13-39aab03bfee864bf-e616994e29cae238
  SK_ar: 7ef28f030c54ef5a-84c9765946e408da-d026e7294a209973-3d607b182eb86ec4
  PPK: no
  message-id sent/recv: 2/0
  QKD: no
  PQC-KEM (IKE): no
  PQC-KEM (all IPsec): no
  lifetime/rekey: 86400/85722
  DPD sent/recv: 00000000/00000000
  peer-id: 172.16.200.4
```

2. Confirm that FEC is enabled in the egress direction for the tohub-d tunnel:

```
# diagnose vpn tunnel list
list all ipsec tunnel in vd 0
------------------------------------------------------
name=tohub-d_isp2 ver=2 serial=5 11.101.1.1:5929->172.16.200.4:4500 nexthop=11.101.1.2 tun_
id=10.0.0.2 tun_id6=::10.0.0.2 status=up dst_mtu=1500 weight=1 country=ZZ
bound_if=15 real_if=15 lgwy=static/1 tun=intf mode=auto/1 encap=none options[0x228]=npu frag-
rfc  run_state=0 role=primary accept_traffic=1 overlay_id=0

proxyid_num=1 child_num=0 refcnt=5 ilast=31 olast=31 ad=r/2
stat: rxp=347 txp=22 rxb=458878 txb=1410
dpd: mode=on-demand on=1 status=ok idle=20000ms retry=3 count=0 seqno=1802
natt: mode=none draft=0 interval=0 remote_port=0
proxyid=tohub-d_isp2 proto=0 sa=1 ref=3 serial=1 auto-negotiate adr
  src: 0:0.0.0.0-255.255.255.255:0
```

```
  dst: 0:0.0.0.0-255.255.255.255:0
  SA:  ref=3 options=1a227 type=00 soft=0 mtu=1438 expire=42519/0B replaywin=2048
       seqno=17 esn=0 replaywin_lastseq=0000015c qat=0 rekey=0 hash_search_len=1
  life: type=01 bytes=0/0 timeout=42902/43200
  dec: spi=18e0b19e esp=aes key=16 c005a5c587fcd4ad85686868323f7a21
       ah=sha256 key=32 3da0b40b525f363355acb3dd445f0131c924b14bb3f704dfd8d7b205412e01d9
  enc: spi=c0e46a65 esp=aes key=16 a161fb71279cd02bed75614c16ce48d0
       ah=sha256 key=32 34977a64e77716c20d4fdfb4619f0745f8d850e448e0330a64baed64dbc24f91
  dec:pkts/bytes=347/458878, enc:pkts/bytes=22/2952
  npu_flag=00 npu_rgwy=0.0.0.0:0 npu_lgwy=0.0.0.0:0npu_selid=6
  dec_npuid=0 enc_npuid=0 dec_engid=-1 enc_engid=-1 dec_saidx=-1 enc_saidx=-1
------------------------------------------------------
name=tohub-d ver=2 serial=4 172.16.200.1:0->172.16.200.4:0 nexthop=172.16.200.4 tun_
id=172.16.200.4 tun_id6=::172.16.200.4 status=up dst_mtu=1500 weight=1 country=ZZ
bound_if=7 real_if=7 lgwy=static/1 tun=intf mode=auto/1 encap=none options[0x228]=npu frag-rfc
 run_state=0 role=primary accept_traffic=1 overlay_id=0

proxyid_num=1 child_num=0 refcnt=4 ilast=1 olast=1 ad=r/2
stat: rxp=375 txp=937 rxb=40092 txb=61856
dpd: mode=on-demand on=1 status=ok idle=20000ms retry=3 count=0 seqno=16
natt: mode=none draft=0 interval=0 remote_port=0
fec: egress=1 ingress=0
proxyid=tohub-d proto=0 sa=1 ref=31 serial=1 auto-negotiate adr
  src: 0:0.0.0.0-255.255.255.255:0
  dst: 0:0.0.0.0-255.255.255.255:0
  SA:  ref=6 options=1a227 type=00 soft=0 mtu=1342 expire=42509/0B replaywin=2048
       seqno=17aa esn=0 replaywin_lastseq=0000004d qat=0 rekey=0 hash_search_len=1
  life: type=01 bytes=0/0 timeout=42898/43200
  dec: spi=18e0b19f esp=aes key=16 3e578ce917adbf940e9ad602baa71a92
       ah=sha256 key=32 4a23b8861918b8ad336cae572dd17f808e64ee22367a219d00249d06405736cf
  enc: spi=c0e46a64 esp=aes key=16 b5e7464c94b4847c30c804f7effa8a0d
       ah=sha256 key=32 bbde5831078147c8b3fd88d0d5112b6110cce75bdb5b23392cc3c2f90c9e4336
  dec:pkts/bytes=6/240, enc:pkts/bytes=937/125532
  npu_flag=03 npu_rgwy=172.16.200.4:0 npu_lgwy=172.16.200.1:0npu_selid=5
  dec_npuid=1 enc_npuid=1 dec_engid=-1 enc_engid=-1 dec_saidx=7399 enc_saidx=4
```

3. Confirm that the profile selected is sending three parity packets per nine base packets:

```
# diagnose vpn tunnel fec tohub-d
egress:  enabled=1
        base=9 redundant=3 codec=0 timeout=10(ms)
        encode=132 encode_timeout=129 encode_fail=0
        tx_data=166 tx_parity=396
        total_tx_data=166 total_tx_parity=396
ingress: enabled=0
        timeout=0(ms) fasm_cnt=0 fasm_full=0
        ipsec_fec_chk_fail=0 complete=0
        recover=0 recover_timeout=0 recover_fail=0
        rx_data=0 rx_parity=0
        rx=0 rx_fail=0
```

ToS-based SD-WAN duplication matching

Duplication rules now support matching traffic based on ToS (Type of Service), enabling more precise control. Previously, ToS criterion was not available in the duplication configuration.

The `config duplication` command includes new options:

```
config system sdwan
    config duplication
      edit 1
        ...
        set tos <8bit hex value>
        set tos-mask <8bit hex value>
      next
    end
end
```

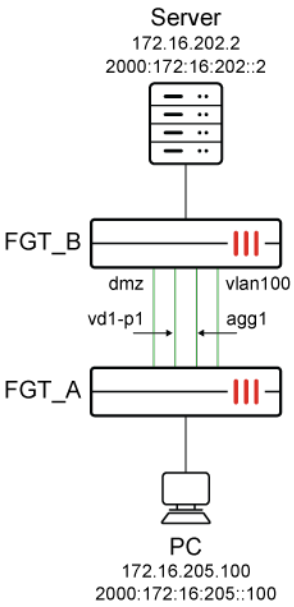
Option	Description
`tos <8bit hex value>`	Type of service bit pattern. The 8bit hex value is the bit pattern to match after applying `tos-mask`.
`tos-mask <8bit hex value>`	Type of service evaluated bits. The 8bit hex value is the bit mask of bytes to compare to ToS.

Example

This example demonstrates how to configure duplication and de-duplication rules that match traffic based on ToS 0x77.

FGT_A is configured with a zone (zone1) that allows a maximum of four zone members (dmz, vd1-p1, agg1, and vlan100) to perform duplication on packets with ToS 0x77.

FGT_B is configured with a zone (virtual-wan-link) that allows four zone members (dmz, vd1-p1, agg1, and vlan100) to perform de-duplication on packets with ToS 0x77.



Configuring duplication on FGT_A

To configure duplication on FGT_A:

1. On FGT_A, enable SD-WAN, set a duplication maximum, create a zone with four members, and create a duplication rule for ToS 0x77:

   A duplication rule is defined to only match packets with ToS 0x77.

```
config system sdwan
    set status enable
    set duplication-max-num 4
    config zone
        edit "virtual-wan-link"
        next
        edit "zone1"
        next
    end
    config members
        edit 1
            set interface "dmz"
            set zone "zone1"
            set gateway 172.16.208.2
            set gateway6 2000:172:16:208::2
        next
        edit 2
            set interface "vd1-p1"
            set zone "zone1"
        next
        edit 3
            set interface "agg1"
            set zone "zone1"
```

```
            set gateway 172.16.203.2
            set gateway6 2000:172:16:203::2
        next
        edit 4
            set interface "vlan100"
            set zone "zone1"
            set gateway 172.16.206.2
            set gateway6 2000:172:16:206::2
        next
    end
    config duplication
        edit 1
            set srcaddr "172.16.205.0"
            set dstaddr "172.16.202.0"
            set srcaddr6 "2000:172:16:205::"
            set dstaddr6 "2000:172:16:202::"
            set srcintf "port5"
            set dstintf "zzzzzzzzzzz"
            set service "PING" "PING6"
            set tos 0x77
            set tos-mask 0xff
            set packet-duplication force
        next
    end
end
```

2. Configure IPv4 and IPv6 routes on members of zone1:

```
config router static
    ...
    edit 2
        set distance 1
        set sdwan-zone "virtual-wan-link" "zone1"
    next
end

config router static6
    ...
    edit 2
        set distance 1
        set sdwan-zone "virtual-wan-link" "zone1"
    next
end
```

3. Check IPv4 and IPv6 routing on which duplication relies:

```
get router info routing-table static
Routing table for VRF=0
S*      0.0.0.0/0 [1/0] via 172.16.203.2, agg1, [1/0]
                        [1/0] via 172.16.206.2, vlan100, [1/0]
                        [1/0] via 172.16.208.2, dmz, [1/0]
                        [1/0] via vd1-p1 tunnel 172.16.209.2, [1/0]
```

```
FGT_A (root) (Interim)# get router info6 routing-table static
Routing table for VRF=0
S*      ::/0 [1/0] via vd1-p1 tunnel ::172.16.209.2, 23:20:44, [1024/0]
                [1/0] via 2000:172:16:203::2, agg1, 23:20:44, [1024/0]
                [1/0] via 2000:172:16:206::2, vlan100, 23:20:44, [1024/0]
                [1/0] via 2000:172:16:208::2, dmz, 23:20:44, [1024/0]
```

4. Perform IPv4 and IPv6 ping with ToS 0x00 from PC to server to confirm duplication doesn't occur because duplication rule only matches ToS 0x77:

```
~$ ping 172.16.202.2 -Q 0x00
PING 172.16.202.2 (172.16.202.2) 56(84) bytes of data.
64 bytes from 172.16.202.2: icmp_seq=1 ttl=253 time=0.480 ms
64 bytes from 172.16.202.2: icmp_seq=2 ttl=253 time=0.307 ms
64 bytes from 172.16.202.2: icmp_seq=3 ttl=253 time=0.158 ms
64 bytes from 172.16.202.2: icmp_seq=4 ttl=253 time=0.168 ms
^C
--- 172.16.202.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3059ms
rtt min/avg/max/mdev = 0.158/0.278/0.480/0.131 ms


~$ ping 2000:172:16:202::2 -Q 0x00
PING 2000:172:16:202::2(2000:172:16:202::2) 56 data bytes
64 bytes from 2000:172:16:202::2: icmp_seq=1 ttl=62 time=0.699 ms
64 bytes from 2000:172:16:202::2: icmp_seq=2 ttl=62 time=0.416 ms
64 bytes from 2000:172:16:202::2: icmp_seq=3 ttl=62 time=0.178 ms
64 bytes from 2000:172:16:202::2: icmp_seq=4 ttl=62 time=0.179 ms
^C
--- 2000:172:16:202::2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3051ms
rtt min/avg/max/mdev = 0.178/0.368/0.699/0.214 ms
```

5. Perform IPv4 and IPv6 ping with ToS 0x77 from PC to server to confirm duplication occurs:

```
~$ ping 172.16.202.2 -Q 0x77
PING 172.16.202.2 (172.16.202.2) 56(84) bytes of data.
64 bytes from 172.16.202.2: icmp_seq=1 ttl=253 time=0.609 ms
64 bytes from 172.16.202.2: icmp_seq=1 ttl=253 time=0.609 ms (DUP!)
64 bytes from 172.16.202.2: icmp_seq=1 ttl=253 time=0.609 ms (DUP!)
64 bytes from 172.16.202.2: icmp_seq=1 ttl=253 time=0.609 ms (DUP!)
64 bytes from 172.16.202.2: icmp_seq=2 ttl=253 time=0.512 ms
64 bytes from 172.16.202.2: icmp_seq=2 ttl=253 time=0.512 ms (DUP!)
64 bytes from 172.16.202.2: icmp_seq=2 ttl=253 time=0.512 ms (DUP!)
64 bytes from 172.16.202.2: icmp_seq=2 ttl=253 time=0.512 ms (DUP!)
^C
--- 172.16.202.2 ping statistics ---
2 packets transmitted, 2 received, +6 duplicates, 0% packet loss, time 1005ms
rtt min/avg/max/mdev = 0.512/0.560/0.609/0.053 ms
```

```
~$ ping 2000:172:16:202::2 -Q 0x77
PING 2000:172:16:202::2(2000:172:16:202::2) 56 data bytes
64 bytes from 2000:172:16:202::2: icmp_seq=1 ttl=62 time=0.551 ms
64 bytes from 2000:172:16:202::2: icmp_seq=1 ttl=62 time=0.551 ms (DUP!)
64 bytes from 2000:172:16:202::2: icmp_seq=1 ttl=62 time=0.551 ms (DUP!)
64 bytes from 2000:172:16:202::2: icmp_seq=1 ttl=62 time=0.551 ms (DUP!)
64 bytes from 2000:172:16:202::2: icmp_seq=2 ttl=62 time=0.429 ms
64 bytes from 2000:172:16:202::2: icmp_seq=2 ttl=62 time=0.429 ms (DUP!)
64 bytes from 2000:172:16:202::2: icmp_seq=2 ttl=62 time=0.429 ms (DUP!)
64 bytes from 2000:172:16:202::2: icmp_seq=2 ttl=62 time=0.429 ms (DUP!)
^C
--- 2000:172:16:202::2 ping statistics ---
2 packets transmitted, 2 received, +6 duplicates, 0% packet loss, time 1010ms
rtt min/avg/max/mdev = 0.429/0.490/0.551/0.061 ms
```

Configuring de-duplication on FGT_B

To configure de-duplication on FGT_B:

1. On FGT_B, enable SD-WAN, define zone members, and define a de-duplication rule:
   A de-duplication rule is defined to only match packets with ToS 0x77.

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "dmz"
            set gateway 172.16.208.1
            set gateway6 2000:172:16:208::1
        next
        edit 2
            set interface "vd1-p1"
        next
        edit 3
            set interface "agg1"
            set gateway 172.16.203.1
            set gateway6 2000:172:16:203::1
        next
        edit 4
            set interface "vlan100"
            set gateway 172.16.206.1
            set gateway6 2000:172:16:206::1
        next
    end
    config duplication
        edit 1
```

```
            set srcaddr "172.16.205.0"
            set dstaddr "172.16.202.0"
            set srcaddr6 "2000:172:16:205::"
            set dstaddr6 "2000:172:16:202::"
            set srcintf "virtual-wan-link"
            set dstintf "port5"
            set service "PING" "PING6"
            set tos 0x77
            set tos-mask 0xff
            set packet-de-duplication enable
        next
    end
end
```

2. Perform an IPv4 and IPv6 ping with ToS 0x77 from PC to server to show that de-duplication occurs on FGT_B:

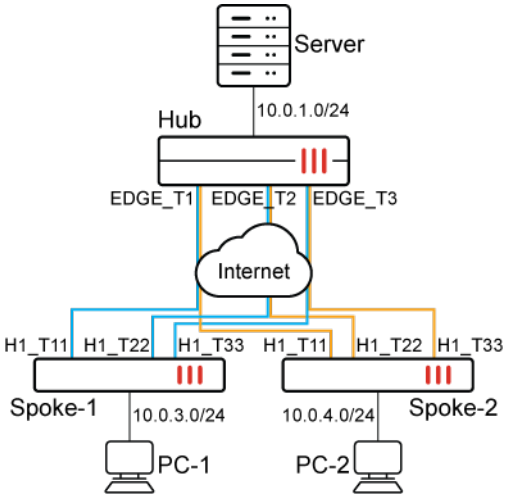
```
~$ ping 172.16.202.2 -Q 0x77
PING 172.16.202.2 (172.16.202.2) 56(84) bytes of data.
64 bytes from 172.16.202.2: icmp_seq=1 ttl=253 time=0.587 ms
64 bytes from 172.16.202.2: icmp_seq=2 ttl=253 time=0.451 ms
64 bytes from 172.16.202.2: icmp_seq=3 ttl=253 time=0.458 ms
64 bytes from 172.16.202.2: icmp_seq=4 ttl=253 time=0.428 ms
64 bytes from 172.16.202.2: icmp_seq=5 ttl=253 time=0.437 ms
^C
--- 172.16.202.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4074ms
rtt min/avg/max/mdev = 0.428/0.472/0.587/0.059 ms

~$ ping 2000:172:16:202::2 -Q 0x77
PING 2000:172:16:202::2(2000:172:16:202::2) 56 data bytes
64 bytes from 2000:172:16:202::2: icmp_seq=1 ttl=62 time=0.758 ms
64 bytes from 2000:172:16:202::2: icmp_seq=2 ttl=62 time=0.420 ms
64 bytes from 2000:172:16:202::2: icmp_seq=3 ttl=62 time=0.421 ms
64 bytes from 2000:172:16:202::2: icmp_seq=4 ttl=62 time=0.400 ms
64 bytes from 2000:172:16:202::2: icmp_seq=5 ttl=62 time=0.426 ms
^C
--- 2000:172:16:202::2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4090ms
rtt min/avg/max/mdev = 0.400/0.485/0.758/0.136 ms
```

On-demand duplication at the Hub using remote health-check

Packet duplication on hub devices can be configured to automatically activate when a spoke begins sending out-of-SLA remote health checks. Duplication continues until the hub once again receives in-SLA health checks from that spoke. The duplication is limited to the affected spoke's tunnels. For each original packet, the hub sends one

duplicate per tunnel until either all tunnels have transmitted the same packet or the configured maximum (`duplication-max-num`) is reached.



In this example, the Hub has three dial-up tunnels to Spoke-1 and Spoke-2. On-demand duplication is configured on the Hub, and active health-check is configured on the spokes to detect the overlays' SLA.

To configure the Hub:

```
config system sdwan
    set status enable
    set duplication-max-num 4
    config zone
        ...
        edit "overlay"
        next
    end
    config members
        edit 1
            set interface "EDGE_T1"
            set zone "overlay"
        next
        edit 2
            set interface "EDGE_T2"
            set zone "overlay"
        next
        edit 3
            set interface "EDGE_T3"
            set zone "overlay"
        next
       ...
    end
    config health-check
        edit "passive_hc"
            set detect-mode remote
            set sla-id-redistribute 1
            set members 1 2 3
```

```
            config sla
                edit 1
                    set link-cost-factor remote
                next
            end
        next
    end
    ...
    config duplication
        edit 1
            set srcaddr "CORP_LAN"
            set dstaddr "CORP_LAN"
            set srcintf "lan_zone" "overlay"
            set dstintf "overlay" "lan_zone"
            set service "PING"
            set packet-duplication on-demand
        next
    end
end
```

To configure Spoke-1:

```
config system sdwan
    set status enable
    config zone
        ...
        edit "overlay"
        next
    end
    config members
        ...
        edit 4
            set interface "H1_T11"
            set zone "overlay"
            set source 172.31.0.65
            set priority 10
        next
        edit 5
            set interface "H1_T22"
            set zone "overlay"
            set source 172.31.0.65
            set priority 10
        next
        edit 6
            set interface "H1_T33"
            set zone "overlay"
            set source 172.31.0.65
            set priority 10
        next
        ...
    end
    config health-check
```

```
        edit "HUB"
            set server "172.31.100.100"
            set embed-measured-health enable
            set sla-id-redistribute 1
            set members 4 5 6
            config sla
                edit 1
                    set link-cost-factor latency
                    set latency-threshold 100
                next
            end
        next
    end
    ...
end
```

To configure Spoke-2:

```
config system sdwan
    set status enable
    config zone
    ...
        edit "overlay"
    end
    config members
        edit 4
            set interface "H1_T11"
            set zone "overlay"
            set source 172.31.0.66
            set cost 2
            set priority 10
        next
        edit 5
            set interface "H1_T22"
            set zone "overlay"
            set source 172.31.0.66
            set priority 10
        next
        edit 6
            set interface "H1_T33"
            set zone "overlay"
            set source 172.31.0.66
            set priority 10
        next
    ...
    end
    config health-check
        edit "HUB"
            set server "172.31.100.100"
            set embed-measured-health enable
            set sla-id-redistribute 1
            set members 4 5 6
```

```
            config sla
                edit 1
                    set link-cost-factor latency
                    set latency-threshold 200
                next
            end
        next
    end
    ...
end
```

To check the routes:

1. On the Hub, check the subnets behind Spoke-1 (10.0.3.0/24) and Spoke-2 (10.0.4.0/24):

```
# get router info routing-table details 10.0.3.0/24

Routing table for VRF=0
Routing entry for 10.0.3.0/24
  Known via "bgp", distance 200, metric 0, best
  Last update 01:10:23 ago
  * vrf 0 172.31.0.65 priority 1 (recursive via EDGE_T1 tunnel 172.31.0.65 [1])
                                 (recursive via EDGE_T2 tunnel 10.0.0.14 [1])
                                 (recursive via EDGE_T3 tunnel 10.0.0.15 [1])
```

```
# get router info routing-table details 10.0.4.0/24

Routing table for VRF=0
Routing entry for 10.0.4.0/24
  Known via "bgp", distance 200, metric 0, best
  Last update 01:09:45 ago
  * vrf 0 172.31.0.66 priority 1 (recursive via EDGE_T1 tunnel 172.31.0.66 [1])
                                 (recursive via EDGE_T2 tunnel 10.0.0.16 [1])
                                 (recursive via EDGE_T3 tunnel 10.0.0.17 [1])
```

2. On Spoke-1, check the subnet behind the Hub:

```
# get router info routing-table details 10.0.1.0/24

Routing table for VRF=0
Routing entry for 10.0.1.0/24
  Known via "bgp", distance 200, metric 0, best
  Last update 01:40:18 ago
  * vrf 0 172.31.0.1, tag 1 priority 1 (recursive via H1_T11 tunnel 172.31.1.1), best-match
                                       (recursive via H1_T22 tunnel 172.31.1.5), best-match
                                       (recursive via H1_T33 tunnel 172.31.2.1), best-match
```

3. On Spoke-2, check the subnet behind the Hub:

```
# get router info routing-table details 10.0.1.0/24

Routing table for VRF=0
```

```
Routing entry for 10.0.1.0/24
  Known via "bgp", distance 200, metric 0, best
  Last update 01:11:37 ago
  * vrf 0 172.31.0.1, tag 1 priority 1 (recursive via H1_T11 tunnel 172.31.1.1), best-match
                                       (recursive via H1_T22 tunnel 172.31.1.5), best-match
                                       (recursive via H1_T33 tunnel 172.31.2.1), best-match
```

4. On the Hub, check the remote health status and that all child tunnels are in-SLA:

```
# diagnose sys sdwan health-check remote
Remote Health Check: passive_hc(3)
  Passive remote statistics of EDGE_T3(47):
EDGE_T3_0(10.0.0.15): timestamp=02-19 12:10:03.556, src=172.31.0.65, latency=0.150,
jitter=0.007, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_prio=0,
last_sla_change=02-19 10:23:45.877
EDGE_T3_1(10.0.0.17): timestamp=02-19 12:10:03.368, src=172.31.0.66, latency=0.159,
jitter=0.022, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_prio=0,
last_sla_change=02-19 10:51:17.377
Remote Health Check: passive_hc(2)
  Passive remote statistics of EDGE_T2(46):
EDGE_T2_1(10.0.0.14): timestamp=02-19 12:10:03.556, src=172.31.0.65, latency=0.210,
jitter=0.004, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_prio=0,
last_sla_change=02-19 10:21:44.967
EDGE_T2_0(10.0.0.16): timestamp=02-19 12:10:03.368, src=172.31.0.66, latency=0.229,
jitter=0.017, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_prio=0,
last_sla_change=02-19 10:51:08.667
Remote Health Check: passive_hc(1)
  Passive remote statistics of EDGE_T1(45):
EDGE_T1_1(172.31.0.65): timestamp=02-19 12:10:03.556, src=172.31.0.65, latency=0.262,
jitter=0.019, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_prio=0,
last_sla_change=02-19 12:05:34.137
EDGE_T1_0(172.31.0.66): timestamp=02-19 12:10:03.368, src=172.31.0.66, latency=0.285,
jitter=0.021, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_prio=0,
last_sla_change=02-19 10:51:08.157
```

5. Ping from the server to PC-1 and PC-2 and on the Hub, confirm that traffic goes on EDGE_T1 and no duplication occurs because the child tunnels of EDGE_T1 are in-SLA.
   a. PC-1:

```
test@server:~$ ping 10.0.3.2
PING 10.0.3.2 (10.0.3.2) 56(84) bytes of data.
64 bytes from 10.0.3.2: icmp_seq=1 ttl=62 time=1.09 ms
64 bytes from 10.0.3.2: icmp_seq=2 ttl=62 time=0.684 ms
64 bytes from 10.0.3.2: icmp_seq=3 ttl=62 time=0.679 ms
```

   Packet sniffer on the Hub:

```
# diagnose sniffer packet any 'host 10.0.3.2' 4
interfaces=[any]
filters=[host 10.0.3.2]
39.588602 port4 in 10.0.1.2 -> 10.0.3.2: icmp: echo request
39.588763 EDGE_T1 out 10.0.1.2 -> 10.0.3.2: icmp: echo request
```

```
39.589555 EDGE_T1 in 10.0.3.2 -> 10.0.1.2: icmp: echo reply
39.589626 port4 out 10.0.3.2 -> 10.0.1.2: icmp: echo reply
40.588399 port4 in 10.0.1.2 -> 10.0.3.2: icmp: echo request
40.588472 EDGE_T1 out 10.0.1.2 -> 10.0.3.2: icmp: echo request
40.588946 EDGE_T1 in 10.0.3.2 -> 10.0.1.2: icmp: echo reply
40.588998 port4 out 10.0.3.2 -> 10.0.1.2: icmp: echo reply
41.587411 port4 in 10.0.1.2 -> 10.0.3.2: icmp: echo request
41.587491 EDGE_T1 out 10.0.1.2 -> 10.0.3.2: icmp: echo request
41.587972 EDGE_T1 in 10.0.3.2 -> 10.0.1.2: icmp: echo reply
41.588028 port4 out 10.0.3.2 -> 10.0.1.2: icmp: echo reply
```

b. PC-2:

```
test@server:~$ ping 10.0.4.2
PING 10.0.4.2 (10.0.4.2) 56(84) bytes of data.
64 bytes from 10.0.4.2: icmp_seq=1 ttl=62 time=0.959 ms
64 bytes from 10.0.4.2: icmp_seq=2 ttl=62 time=0.630 ms
64 bytes from 10.0.4.2: icmp_seq=3 ttl=62 time=0.650 ms
```

Packet sniffer on the Hub:

```
# diagnose sniffer packet any 'host 10.0.4.2' 4
interfaces=[any]
filters=[host 10.0.4.2]
2.598946 port4 in 10.0.1.2 -> 10.0.4.2: icmp: echo request
2.599055 EDGE_T1 out 10.0.1.2 -> 10.0.4.2: icmp: echo request
2.599777 EDGE_T1 in 10.0.4.2 -> 10.0.1.2: icmp: echo reply
2.599810 port4 out 10.0.4.2 -> 10.0.1.2: icmp: echo reply
3.598778 port4 in 10.0.1.2 -> 10.0.4.2: icmp: echo request
3.598819 EDGE_T1 out 10.0.1.2 -> 10.0.4.2: icmp: echo request
3.599262 EDGE_T1 in 10.0.4.2 -> 10.0.1.2: icmp: echo reply
3.599295 port4 out 10.0.4.2 -> 10.0.1.2: icmp: echo reply
4.598782 port4 in 10.0.1.2 -> 10.0.4.2: icmp: echo request
4.598819 EDGE_T1 out 10.0.1.2 -> 10.0.4.2: icmp: echo request
4.599279 EDGE_T1 in 10.0.4.2 -> 10.0.1.2: icmp: echo reply
4.599316 port4 out 10.0.4.2 -> 10.0.1.2: icmp: echo reply
```

6. Increase the latency to 120ms on overlay H1_T11 of Spoke-1, then confirm that the corresponding child tunnel on the Hub is out-of-SLA. Also confirm that duplication occurs from the Hub to Spoke-1.

a. Check the health status on the HUB:

```
# diagnose sys sdwan health-check  remote
Remote Health Check: passive_hc(3)
  Passive remote statistics of EDGE_T3(47):
EDGE_T3_0(10.0.0.15): timestamp=02-19 13:28:14.680, src=172.31.0.65, latency=0.158,
jitter=0.009, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_
prio=0, last_sla_change=02-19 10:23:45.877
EDGE_T3_1(10.0.0.17): timestamp=02-19 13:28:14.504, src=172.31.0.66, latency=0.149,
jitter=0.019, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_
prio=0, last_sla_change=02-19 10:51:17.377
Remote Health Check: passive_hc(2)
  Passive remote statistics of EDGE_T2(46):
```

```
EDGE_T2_1(10.0.0.14): timestamp=02-19 13:28:14.680, src=172.31.0.65, latency=0.220,
jitter=0.009, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_
prio=0, last_sla_change=02-19 10:21:44.967
EDGE_T2_0(10.0.0.16): timestamp=02-19 13:28:14.504, src=172.31.0.66, latency=0.213,
jitter=0.014, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_
prio=0, last_sla_change=02-19 10:51:08.667
Remote Health Check: passive_hc(1)
  Passive remote statistics of EDGE_T1(45):
EDGE_T1_1(172.31.0.65): timestamp=02-19 13:28:14.680, src=172.31.0.65, latency=120.247,
jitter=0.017, pktloss=0.000%, mos=4.338, SLA id=1(remote), rmt_ver=1, rmt_sla=out, rmt_
prio=0, last_sla_change=02-19 13:25:41.227
EDGE_T1_0(172.31.0.66): timestamp=02-19 13:28:14.504, src=172.31.0.66, latency=0.255,
jitter=0.031, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_
prio=0, last_sla_change=02-19 10:51:08.157
```

b. Ping PC-1:

The duplication takes place from Hub to Spoke-1.

```
test@server:~$ ping 10.0.3.2

64 bytes from 10.0.3.2: icmp_seq=51 ttl=62 time=120 ms
64 bytes from 10.0.3.2: icmp_seq=52 ttl=62 time=120 ms
64 bytes from 10.0.3.2: icmp_seq=53 ttl=62 time=120 ms
64 bytes from 10.0.3.2: icmp_seq=54 ttl=62 time=0.899 ms
64 bytes from 10.0.3.2: icmp_seq=54 ttl=62 time=0.931 ms (DUP!)
64 bytes from 10.0.3.2: icmp_seq=54 ttl=62 time=120 ms (DUP!)
64 bytes from 10.0.3.2: icmp_seq=55 ttl=62 time=0.820 ms
64 bytes from 10.0.3.2: icmp_seq=55 ttl=62 time=0.855 ms (DUP!)
64 bytes from 10.0.3.2: icmp_seq=55 ttl=62 time=120 ms (DUP!)
64 bytes from 10.0.3.2: icmp_seq=56 ttl=62 time=0.793 ms
64 bytes from 10.0.3.2: icmp_seq=56 ttl=62 time=0.827 ms (DUP!)
64 bytes from 10.0.3.2: icmp_seq=56 ttl=62 time=120 ms (DUP!)
```

Because one tunnel is out-of-SLA, the ping test changes from single echo replies (as indicated by the sequence number increasing by one and the latency remaining at 120ms) to three echo replies per sequence and only one with 120ms latency, indicating that all tunnels are used at once. Note that the duplication was configured to send four duplicate packets, but only two duplicates are received as a result of only three eligible tunnels.

7. Increase the latency to 220ms on overlay H1_T11 of Spoke-2, then confirm that corresponding child tunnel on the Hub is out-of-SLA. Also confirm that duplication occurs from the Hub to Spoke-2.

a. Check the health status on the HUB:

```
# diagnose sys sdwan health-check  remote
Remote Health Check: passive_hc(3)
  Passive remote statistics of EDGE_T3(47):
EDGE_T3_0(10.0.0.15): timestamp=02-19 13:40:38.693, src=172.31.0.65, latency=0.157,
jitter=0.008, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_
prio=0, last_sla_change=02-19 10:23:45.877
EDGE_T3_1(10.0.0.17): timestamp=02-19 13:40:38.766, src=172.31.0.66, latency=0.145,
jitter=0.017, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_
prio=0, last_sla_change=02-19 10:51:17.377
```

```
Remote Health Check: passive_hc(2)
  Passive remote statistics of EDGE_T2(46):
EDGE_T2_1(10.0.0.14): timestamp=02-19 13:40:38.692, src=172.31.0.65, latency=0.220,
jitter=0.006, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_
prio=0, last_sla_change=02-19 10:21:44.967
EDGE_T2_0(10.0.0.16): timestamp=02-19 13:40:38.766, src=172.31.0.66, latency=0.219,
jitter=0.012, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_
prio=0, last_sla_change=02-19 10:51:08.667
Remote Health Check: passive_hc(1)
  Passive remote statistics of EDGE_T1(45):
EDGE_T1_1(172.31.0.65): timestamp=02-19 13:40:38.692, src=172.31.0.65, latency=120.245,
jitter=0.010, pktloss=0.000%, mos=4.338, SLA id=1(remote), rmt_ver=1, rmt_sla=out, rmt_
prio=0, last_sla_change=02-19 13:25:41.227
EDGE_T1_0(172.31.0.66): timestamp=02-19 13:40:38.766, src=172.31.0.66, latency=220.242,
jitter=0.018, pktloss=0.000%, mos=4.103, SLA id=1(remote), rmt_ver=1, rmt_sla=out, rmt_
prio=0, last_sla_change=02-19 13:39:12.427
```

b. Ping PC-2:

The duplication takes place from Hub to Spoke-2.

```
test@server:~$ ping 10.0.4.2

64 bytes from 10.0.4.2: icmp_seq=103 ttl=62 time=220 ms
64 bytes from 10.0.4.2: icmp_seq=104 ttl=62 time=220 ms
64 bytes from 10.0.4.2: icmp_seq=105 ttl=62 time=220 ms
64 bytes from 10.0.4.2: icmp_seq=106 ttl=62 time=0.735 ms
64 bytes from 10.0.4.2: icmp_seq=106 ttl=62 time=0.742 ms (DUP!)
64 bytes from 10.0.4.2: icmp_seq=106 ttl=62 time=220 ms (DUP!)
64 bytes from 10.0.4.2: icmp_seq=107 ttl=62 time=0.721 ms
64 bytes from 10.0.4.2: icmp_seq=107 ttl=62 time=0.729 ms (DUP!)
64 bytes from 10.0.4.2: icmp_seq=107 ttl=62 time=220 ms (DUP!)
64 bytes from 10.0.4.2: icmp_seq=108 ttl=62 time=0.719 ms
64 bytes from 10.0.4.2: icmp_seq=108 ttl=62 time=0.731 ms (DUP!)
64 bytes from 10.0.4.2: icmp_seq=108 ttl=62 time=220 ms (DUP!)
```

8. Remove the latency impairment and see that the duplication stops.

a. Check the health status on the HUB:

```
# diagnose sys sdwan health-check  remote
Remote Health Check: passive_hc(3)
  Passive remote statistics of EDGE_T3(47):
EDGE_T3_0(10.0.0.15): timestamp=02-19 13:44:03.213, src=172.31.0.65, latency=0.156,
jitter=0.013, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_
prio=0, last_sla_change=02-19 10:23:45.877
EDGE_T3_1(10.0.0.17): timestamp=02-19 13:44:03.307, src=172.31.0.66, latency=0.151,
jitter=0.022, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_
prio=0, last_sla_change=02-19 10:51:17.377
Remote Health Check: passive_hc(2)
  Passive remote statistics of EDGE_T2(46):
EDGE_T2_1(10.0.0.14): timestamp=02-19 13:44:03.213, src=172.31.0.65, latency=0.211,
jitter=0.008, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_
prio=0, last_sla_change=02-19 10:21:44.967
```

```
EDGE_T2_0(10.0.0.16): timestamp=02-19 13:44:03.307, src=172.31.0.66, latency=0.224,
jitter=0.016, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_
prio=0, last_sla_change=02-19 10:51:08.667
Remote Health Check: passive_hc(1)
  Passive remote statistics of EDGE_T1(45):
EDGE_T1_1(172.31.0.65): timestamp=02-19 13:44:03.213, src=172.31.0.65, latency=0.269,
jitter=0.028, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_
prio=0, last_sla_change=02-19 13:43:36.967
EDGE_T1_0(172.31.0.66): timestamp=02-19 13:44:03.307, src=172.31.0.66, latency=0.263,
jitter=0.036, pktloss=0.000%, mos=4.404, SLA id=1(pass), rmt_ver=1, rmt_sla=in, rmt_
prio=0, last_sla_change=02-19 13:43:35.947
```

b. Ping PC-1:

Note where the duplication stops.

```
test@server:$ ping 10.0.3.2

64 bytes from 10.0.3.2: icmp_seq=150 ttl=62 time=0.579 ms
64 bytes from 10.0.3.2: icmp_seq=150 ttl=62 time=0.640 ms (DUP!)
64 bytes from 10.0.3.2: icmp_seq=150 ttl=62 time=0.643 ms (DUP!)
64 bytes from 10.0.3.2: icmp_seq=151 ttl=62 time=0.531 ms
64 bytes from 10.0.3.2: icmp_seq=152 ttl=62 time=0.578 ms
64 bytes from 10.0.3.2: icmp_seq=153 ttl=62 time=0.572 ms
64 bytes from 10.0.3.2: icmp_seq=154 ttl=62 time=0.557 ms
```

c. Ping PC-2:

Note where the duplication stops.

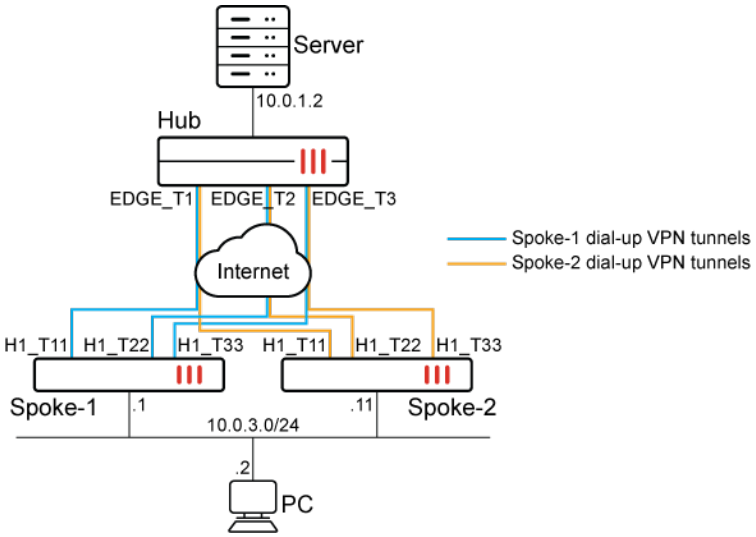
```
test@server:~$ ping 10.0.4.2

64 bytes from 10.0.4.2: icmp_seq=200 ttl=62 time=0.730 ms
64 bytes from 10.0.4.2: icmp_seq=200 ttl=62 time=0.767 ms (DUP!)
64 bytes from 10.0.4.2: icmp_seq=200 ttl=62 time=0.769 ms (DUP!)
64 bytes from 10.0.4.2: icmp_seq=201 ttl=62 time=0.672 ms
64 bytes from 10.0.4.2: icmp_seq=202 ttl=62 time=0.551 ms
64 bytes from 10.0.4.2: icmp_seq=203 ttl=62 time=0.543 ms
64 bytes from 10.0.4.2: icmp_seq=204 ttl=62 time=0.568 ms
```

Confine packet duplication across multiple overlays to the same spoke

Packet duplication can be confined across multiple dial-up tunnels to the same spoke. When the same subnet is behind multiple spokes, a hub device will use the location ID setting to uniquely identify and limit the destination for duplicated traffic.

Example



In this example, the Hub has three dial-up tunnels to Spoke-1 and Spoke-2. Duplication, regardless of SLA status, and de-duplication are enabled on the Hub and spokes, and four copies are allowed for duplication.

Since both spokes are connected to the same LAN subnet (10.0.3.0/24), we can optimize duplication by confining the duplication to the Spoke that initiated the traffic based on its location-id. This example demonstrates that when traffic from the PC is sent through Spoke-1, duplication occurs only on the 3 overlays connected to Spoke-1, even when duplication allows 4 copies to be duplicated.

To configure the Hub:

```
config system sdwan
    set status enable
    set duplication-max-num 4
    config zone
     ...
        edit "overlay"
        next
    end
    config members
        edit 1
            set interface "EDGE_T1"
            set zone "overlay"
        next
        edit 2
            set interface "EDGE_T2"
            set zone "overlay"
        next
        edit 3
            set interface "EDGE_T3"
            set zone "overlay"
        next
    ...
```

```
    end
    config duplication
        edit 1
            set srcaddr "CORP_LAN"
            set dstaddr "CORP_LAN"
            set srcintf "lan_zone" "overlay"
            set dstintf "overlay" "lan_zone"
            set service "PING"
            set packet-duplication force
            set packet-de-duplication enable
        next
    end
end
```

```
config system settings
    set location-id 172.31.0.1
end
```

To configure Spoke-1:

```
config system sdwan
    set status enable
    set duplication-max-num 4
    config zone
    ...
        edit "overlay"
        next
    end
    config members
       ...
        edit 4
            set interface "H1_T11"
            set zone "overlay"
            set source 172.31.0.65
            set priority 10
        next
        edit 5
            set interface "H1_T22"
            set zone "overlay"
            set source 172.31.0.65
            set priority 10
        next
        edit 6
            set interface "H1_T33"
            set zone "overlay"
            set source 172.31.0.65
            set priority 10
        next
      ...
    end
    config duplication
```

```
        edit 1
            set srcaddr "CORP_LAN"
            set dstaddr "CORP_LAN"
            set srcintf "lan_zone" "overlay"
            set dstintf "overlay" "lan_zone"
            set service "PING"
            set packet-duplication force
            set packet-de-duplication enable
        next
    end
end
```

```
config system settings
    set location-id 172.31.0.65
end
```

To configure Spoke-2:

```
config system sdwan
    set status enable
    set duplication-max-num 4
    config zone
    ...
        edit "overlay"
        next
    end
    config members
       ...
        edit 4
            set interface "H1_T11"
            set zone "overlay"
            set source 172.31.0.66
            set priority 10
        next
        edit 5
            set interface "H1_T22"
            set zone "overlay"
            set source 172.31.0.66
            set priority 10
        next
        edit 6
            set interface "H1_T33"
            set zone "overlay"
            set source 172.31.0.66
            set priority 10
        next
      ...
    end
    config duplication
        edit 1
            set srcaddr "CORP_LAN"
```

```
            set dstaddr "CORP_LAN"
            set srcintf "lan_zone" "overlay"
            set dstintf "overlay" "lan_zone"
            set service "PING"
            set packet-duplication force
            set packet-de-duplication enable
        next
    end
end
```

```
config system settings
    set location-id 172.31.0.66
end
```

To check the routes:

1. On the hub, route 10.0.3.0/24 can be learned from both Spoke-1 and Spoke-2 and it is then possible to duplicate traffic to Spoke-1 and Spoke-2, but with this enhancement, traffic is only allowed to be duplicated to the same spoke.

```
# get router info routing-table details 10.0.3.0/24

Routing table for VRF=0
Routing entry for 10.0.3.0/24
  Known via "bgp", distance 200, metric 0, best
  Last update 01:35:35 ago
  * vrf 0 172.31.0.65 priority 1 (recursive via EDGE_T3 tunnel 10.0.0.7 [1])
                                              (recursive via EDGE_T1 tunnel 172.31.0.65 [10])
                                              (recursive via EDGE_T2 tunnel 10.0.0.6 [15])
  * vrf 0 172.31.0.66 priority 1 (recursive via EDGE_T3 tunnel 172.31.0.66 [1])
                                              (recursive via EDGE_T1 tunnel 10.0.0.4 [10])
                                              (recursive via EDGE_T2 tunnel 10.0.0.5 [15])
```

2. On Spoke-1, route 10.0.1.0/24 is learned from the Hub.

```
# get router info routing-table details 10.0.1.0/24

Routing table for VRF=0
Routing entry for 10.0.1.0/24
  Known via "bgp", distance 200, metric 0, best
  Last update 01:52:01 ago
  * vrf 0 172.31.0.1, tag 1 priority 1 (recursive via H1_T11 tunnel 172.31.1.1), best-match
                                                    (recursive via H1_T22 tunnel
172.31.1.5), best-match
                                                    (recursive via H1_T33 tunnel
172.31.2.1), best-match
```

3. From the PC, ping the server through Spoke-1:

```
> ping 10.0.1.2

Pinging 10.0.1.2 with 32 bytes of data:
```

```
Reply from 10.0.1.2: bytes=32 time<1ms TTL=62
Ping statistics for 10.0.1.2:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

4. On Spoke-1, traffic is duplicated to all tunnels (H1_11, H1_T22, and H1_T33), duplicated traffic is received from the Hub, and traffic is de-duplicated then sent to the PC:

```
# diagnose sniffer packet any 'host 10.0.1.2' 4
interfaces=[any]
filters=[host 10.0.1.2]
8.129527 port4 in 10.0.3.2 -> 10.0.1.2: icmp: echo request
8.129913 H1_T22 out 10.0.3.2 -> 10.0.1.2: icmp: echo request                    // traffic is
duplicated to all tunnel H1_11, H1_T22 and H1_T33
8.129939 H1_T33 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
8.129961 H1_T11 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
8.130481 H1_T11 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply                           //
duplicated traffic is received from Hub
8.130504 H1_T33 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
8.130521 H1_T22 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
8.130565 port4 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply                           //
traffic is de-duplicated and sent to PC
```

5. On the Hub, duplicated traffic is received from Spoke-1, traffic is de-duplicated and sent to the server, and traffic is only duplicated to dial-up tunnels destined for Spoke-1 :

```
#  diagnose sniffer packet any 'host 10.0.1.2' 4
interfaces=[any]
filters=[host 10.0.1.2]
3.905902 EDGE_T3 in 10.0.3.2 -> 10.0.1.2: icmp: echo request
3.905910 EDGE_T2 in 10.0.3.2 -> 10.0.1.2: icmp: echo request
3.905951 EDGE_T1 in 10.0.3.2 -> 10.0.1.2: icmp: echo request
3.906110 port4 out 10.0.3.2 -> 10.0.1.2: icmp: echo request
3.906213 port4 in 10.0.1.2 -> 10.0.3.2: icmp: echo reply
3.906295 EDGE_T1 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
3.906332 EDGE_T2 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
3.906360 EDGE_T3 out 10.0.1.2 -> 10.0.3.2: icmp: echo reply
```

6. On Spoke-2, the Hub echo reply packets are not duplicated to dial-up tunnels destined for Spoke-2:

```
# diagnose sniffer packet any 'host 10.0.1.2' 4
interfaces=[any]
filters=[host 10.0.1.2]
```

SD-WAN speed test scheduling improvements

Speed-test scheduling is now coordinated automatically between the FortiGate appliance and the backend Fortinet servers, removing the need to configure retry settings. Using speed-test results for traffic shaping has also

been simplified, with all related settings moved directly into the interface configuration. Speed-test traffic now ignores interface traffic shaping but is affected by values for `update-inbandwidth-maximum` and `update-outbandwidth-maximum` settings when configured within that speed-test's settings. These changes reduce administrative overhead and make speed-test-driven shaping more consistent and easier to manage.

Speed-test scheduling and usage improvement

Speed-test scheduling and usage has been improved in two ways:

- Speed-test settings

  A coordination mechanism between the FortiGate appliance and the backend server has been introduced to streamline speed-test operations. With this enhancement, scheduled speed tests no longer require explicit configuration of `retry` or `retry-pause` parameters, as these behaviors are now managed automatically by the coordination logic. Speed-test traffic no longer uses dedicated shaping settings, and now respects any configured maximum interface bandwidth values for that speed-test-schedule.
- Speed-test result usage

  The configuration required to use speed-test results for interface traffic shaping has been simplified, and settings are now located in the interface configuration.

Affected CLI commands

Speed test settings

This section covers the existing, removed, and new commands for speed test settings.

Existing commands

These commands control maximum and minimum downloading bandwidth values to be considered effective, in Kbps (0 - 16776000). The update-inbandwidth-maximum and update-outbandwidth-maximum settings also function as traffic shaping for speed-tests. When minimums are not met, or maximums are exceeded, the minimum or maximum will be used instead of the measured value.

```
config system speed-test-schedule
    edit <interface>
        set update-inbandwidth-maximum <integer>
        set update-inbandwidth-minimum <integer>
        set update-outbandwidth-maximum <integer>
        set update-outbandwidth-minimum <integer>
```

Removed commands

The `retries` and `retry-pause` commands have been removed and the functionality has been migrated to the FortiGate and Server coordination mechanism.

```
config system speed-test-schedule
    edit <interface>
        set retries <int>
        set retry-pause <int>
```

The `update-inbandwidth` and `update-outbandwidth` commands are no longer used to apply shaping to speed-tests. The existing commands `update-inbandwidth-maximum` and `update-outbandwidth-maximum` are used to apply shaping during speed-tests.

```
config system speed-test-schedule
    edit <interface>
        set update-inbandwidth {enable | disable}
        set update-outbandwidth {enable | disable}
```

The `update-shaper` command has been removed, and the functionality has been replaced by the new command `set inbandwidth-source measured`. See New commands on page 47 for details.

```
config system speed-test-schedule
    edit <interface>
        set update-shaper
```

New commands

The `update-bandwidth-limit-unit` option controls how the values of `update-inbandwidth-maximum/minimum` and `update-outbandwidth-maximum/minimum` are interpreted. By default, these values are treated as kilobits per second (kbps). When set to `percentage`, the values are interpreted as percentages of the interface's configured inbound or outbound bandwidth.

```
config system speed-test-schedule
    edit <interface>
        set update-bandwidth-limit-unit {kbps* | percentage}
```

Speed test results used for traffic shaping

This section covers the existing, removed, and new commands for speed test results used for traffic shaping.

Existing commands

```
config system interface
    set inbandwidth <integer and manually configured>
    set outbandwidth <integer and manually configured>
    set measured-upstream-bandwidth <integer and populated by speed-test results>
    set measured-downstream-bandwidth <integer and populated by speed-test results>
```

Removed commands

The function controlled by the optional `update-int-shaping` field in the `execute speed-test` command is now controlled by the new settings `set inbandwidth-source measured` and `set outbandwidth-source measured`. If

a speed-test is manually executed, these commands will control whether the speed-test results are applied to the interface.

```
execute speed-test ... <update-intf-shaping> ...
```

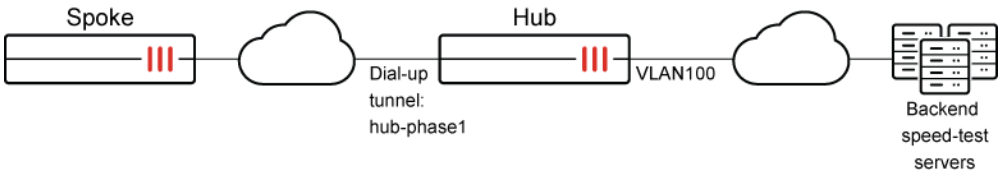
New commands

```
config system interface
    set inbandwidth-source {default* | measured}
    set inbandwidth-source {default* | measured}
```

These commands determine which value from the Existing commands on page 47 takes effect, and replace the following SD-WAN with FortiOS, FortiManager, and FortiAnalyzer 7.6 command:

```
config system speed-test-schedule
    set update-interface-shaping {enable | disable}
```

Example



This section contains the configuration settings for the Hub and Spoke FortiGates to demonstrate two uses of the speed-test:

1. A speed-test is scheduled and run from the Hub to Spoke just before the business opens at 8 am. These results are applied to the traffic shaping on the spoke dial-up tunnel.
2. A second speed-test is run between the Hub and the internet, and the result is applied to the corresponding Hub interface (vlan100 in this example).

Key settings on Hub:

Use speed-test measured bandwidth as inbandwidth:

```
Hub# show sys interface vlan100
config system interface
    edit "vlan100"
        set vdom "root"
        set ip 172.16.206.1 255.255.255.0
        set allowaccess ping https ssh snmp http
        set egress-shaping-profile "profile_1"
        set ingress-shaping-profile "profile_1"
        set inbandwidth-source measured
        set outbandwidth-source measured
        ...
        set vlanid 100
```

```
    next
end
```

Allow `speed-test` protocol on overlay for IPsec speed test. Apply egress-shaping-profile. Use speed-test measured bandwidth as outbandwidth:

```
Hub# show sys interface hub-phase1
config system interface
    edit "hub-phase1"
        set vdom "root"
        set ip 10.10.15.253 255.255.255.255
        set allowaccess ping https ssh snmp http telnet radius-acct probe-response fabric ftm
speed-test
        set type tunnel
        set egress-shaping-profile "profile_1"
        set outbandwidth-source measured
        set remote-ip 10.10.15.254 255.255.255.0
        ...
        set interface "port1"
    next
end
```

Allow `speed-test` protocol on underlay for IPsec speed test.

```
Hub# show sys interface  port1
config system interface
    edit "port1"
        set vdom "root"
        set ip 172.16.200.1 255.255.255.0
        set allowaccess ping https ssh snmp http telnet radius-acct probe-response fabric ftm
speed-test
        ...
    next
end
```

Enable and configure the speed test:

```
Hub# show sys speed-test-schedule
config system speed-test-schedule
    edit "hub-phase1"
        set mode TCP
        set schedules "1"
        set dynamic-server enable //Enable IPsec speed test
        set update-bandwidth-limit-unit value //Set the update bandwidth limit by values in kbps
        set update-inbandwidth-maximum 1000000 //Maximum downloading bandwidth (kbps) of the
interface's inbandwidth to be used in shaping
        set update-inbandwidth-minimum 500000 //Minimum downloading bandwidth (kbps) of the
interface's inbandwidth to be used in shaping
        set update-outbandwidth-maximum 1000000 //Maximum uploading bandwidth (kbps) of the
interface's outbandwidth to be used in shaping
        set update-outbandwidth-minimum 500000 //Minimum uploading bandwidth (kbps) of the
interface's outbandwidth to be used in shaping
    next
```

```
    edit "vlan100"
        set mode TCP
        set schedules "2"
        set legacy-server-mode disable //Utilize the coordination mechanism between Fortigate and
backend server to do internet speed-test
        set update-bandwidth-limit-unit value //Set the update bandwidth limits by values in kbps
        set update-inbandwidth-maximum 1000000 //Maximum downloading bandwidth (kbps) of the
interface's inbandwidth to be used in shaping
        set update-inbandwidth-minimum 500000 //Minimum downloading bandwidth (kbps) of the
interface's inbandwidth to be used in shaping
        set update-outbandwidth-maximum 1000000 //Maximum uploading bandwidth (kbps) of the
interface's outbandwidth to be used in shaping
        set update-outbandwidth-minimum 500000 //Minimum uploading bandwidth (kbps) of the
interface's outbandwidth to be used in shaping
    next
end
```

The first schedule:

```
Hub# show firewall schedule recurring 1
config firewall schedule recurring
    edit "1"
        set start 07:43
        set day sunday monday tuesday wednesday thursday friday saturday
    next
end
```

The second schedule:

```
Hub# show firewall schedule recurring 2
config firewall schedule recurring
    edit "2"
        set start 07:44
        set day sunday monday tuesday wednesday thursday friday saturday
    next
end
```

Key settings on Spoke:

Enable speed-test server for IPsec speed test:

```
Spoke# show sys global
config system global
    ...
    set speedtest-server enable
    ...
end
```

Allow `speed-test` protocol on overlay for IPsec speed test:

```
Spoke# show sys interface  spoke11-p1
config system interface
```

```
    edit "spoke11-p1"
        set vdom "root"
        set allowaccess ping https ssh snmp http telnet radius-acct probe-response fabric ftm
speed-test
        set type tunnel
        ...
        set interface "port1"
    next
end
```

Allow `speed-test` protocol on underlay for IPsec speed test.

```
Spoke# show sys interface  port1
config system interface
    edit "port1"
        set vdom "root"
        set ip 172.16.200.2 255.255.255.0
        set allowaccess ping https ssh snmp http telnet radius-acct probe-response fabric ftm
speed-test
        ...
    next
end
```

Example 1: Hub to Spoke speed test

View the speed-test output by running the following debug command prior to the start of the scheduled test:

```
Hub # diagnose debug application speedtestd -1
```

Recurring firewall schedule 1 begins:

```
Hub# fcron_speedtest_arm_sched()-405: Speed test (0x5561afd510) for hub-phase1 will run in 48 s
fcron_speedtest_arm_sched()-405: Speed test (0x5561afd510) for hub-phase1 will run in 86400 s
fcron_speedtest_ipsec_send_request()-574: root: hub-phase1(hub-phase1_0) try=0 token
request=0.0.0.0:0 -> 10.10.15.1:5200, test= 172.16.200.1:0 -> 172.16.200.2:5201
fcron_sptest_ipsec_on_start()-528: root: (00790005) hub-phase1(hub-phase1_0) server notify start
token=42c4abc4-1292-51f1-07b1-9054c39ee49f
__fork_run_ipsec_test()-987: [15270] Run test 00790005 for 'hub-phase1'(port1) to server
172.16.200.2:5201 (tunnel:hub-phase1_0)
[speedtest(15270)] start uploading test.
[speedtest(15270)] Connecting to host 172.16.200.2, port 5201
[speedtest(15270)] [ 28] local 172.16.200.1 port 14251 connected to 172.16.200.2 port 5201
[speedtest(15270)] [ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[speedtest(15270)] [ 28]   0.00-1.01   sec  83.0 MBytes   686 Mbits/sec    0    134 KBytes
…
[speedtest(15270)] client(recver): bytes_recv=425522920, bytes_sent=426768480, sender_time=5.000,
recver_time=5.001
[speedtest(15270)] client(recver): down_speed:  681 Mbits/sec
[speedtest(15270)]
[speedtest(15270)] speed test Done.
fcron_speedtest_notify_func()-1570: Speed test pid=15270 done
```

```
fcron_speedtest_on_test_finish()-1530: test 0x00790005 for 'hub-phase1' succeed with up=692880,
down=680722
fcron_speedtest_save_results()-1428: Write logs to disk: succ=1, fail=0
fcron_speedtest_sync_results()-1456: Sync cached results to secondary devices.
```

The output from the speed-test debug indicates the speed-test completed with an upload throughput of 692880 kbps. To verify that this value has been applied to the corresponding interface, review the VPN tunnel as follows:

```
Hub# diagnose vpn tunnel list
------------------------------------------------------
name=hub-phase1_0 ver=2 serial=9 172.16.200.1:0->172.16.200.2:0 nexthop=172.16.200.2 tun_
id=10.10.15.1 tun_id6=2000:10:10:15::1 status=up dst_mtu=1500 weight=1 country=ZZ
bound_if=11 real_if=11 lgwy=static/1 tun=intf mode=dial_inst/3 encap=none options[0x22a8]=npu
rgwy-chg frag-rfc  run_state=0 role=primary accept_traffic=1 overlay_id=10
...
egress traffic control:
        bandwidth=692880(kbps) lock_hit=0 default_class=2 n_active_class=3
```

Example 2: Hub to Fortinet server internet speed-test

The speed-test output can be seen by running the following debug command prior to the start of the scheduled test:

```
Hub # diagnose debug application speedtest -1
```

Recurring firewall schedule 2 begins:

```
Hub# diagnose vpn tunnel list
------------------------------------------------------
name=hub-phase1_0 ver=2 serial=9 172.16.200.1:0->172.16.200.2:0 nexthop=172.16.200.2 tun_
id=10.10.15.1 tun_id6=2000:10:10:15::1 status=up dst_mtu=1500 weight=1 country=ZZ
bound_if=11 real_if=11 lgwy=static/1 tun=intf mode=dial_inst/3 encap=none options[0x22a8]=npu
rgwy-chg frag-rfc  run_state=0 role=primary accept_traffic=1 overlay_id=10
...
egress traffic control:
        bandwidth=692880(kbps) lock_hit=0 default_class=2 n_active_class=3


Hub# fcron_speedtest_arm_sched()-405: Speed test (0x5561aa61f0) for vlan100 will run in 33 s
...
[speedtest(15619)] speed test Done.
fcron_speedtest_notify_func()-1570: Speed test pid=15619 done
fcron_speedtest_on_test_finish()-1530: test 0x00630000 for 'vlan100' succeed with up=638101,
down=930393
fcron_sptest_cloud_report_result()-715: test(0x00630000): if=vlan100, server=FTNT_CA_Burnaby
(154.52.1.124:5213), uuid=6a82a804-45b1-477f-b94b-f4a21e17596 report result=0, up=638101,
dw=930393
fcron_sptest_cloud_result_reply()-663: test(0x00630000): if=vlan100, uuid=6a82a804-45b1-477f-b94b-
f4a21e17596 report result succeed (status=2).
```

The output from the speed-test debug indicates the speed-test completed with an upload throughput of 638101 kbps and a download throughput of 930393 kbps. To verify that this value has been applied to the corresponding interface, review the VLAN interface as follows:

```
Hub# diagnose netlink interface list vlan100
if=vlan100 alias= family=00 type=1 index=99 mtu=1500 link=0 master=0
ref=28 state=start present fw_flags=10010400 flags=up broadcast run multicast
Qdisc=noqueue hw_addr=e0:23:ff:3e:7f:eb broadcast_addr=ff:ff:ff:ff:ff:ff
ingress traffic control:
        bandwidth=930393(kbps) lock_hit=0 default_class=2 n_active_class=3
...
egress traffic control:
        bandwidth=638101(kbps) lock_hit=0 default_class=2 n_active_class=3
```

Operations

8.0.0

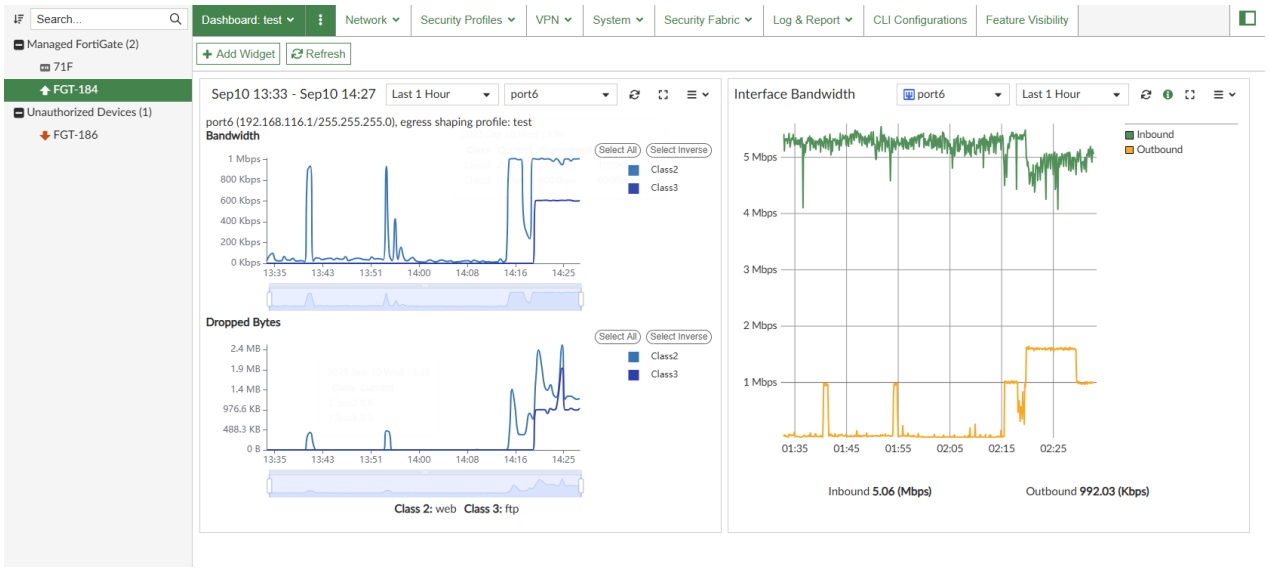
Interface-based bandwidth graph uses average bandwidth logic FMG

The bandwidth graph of the *Traffic Shaping (interface-based)* widget has been enhanced to plot the average bandwidth values instead of the current bandwidth.

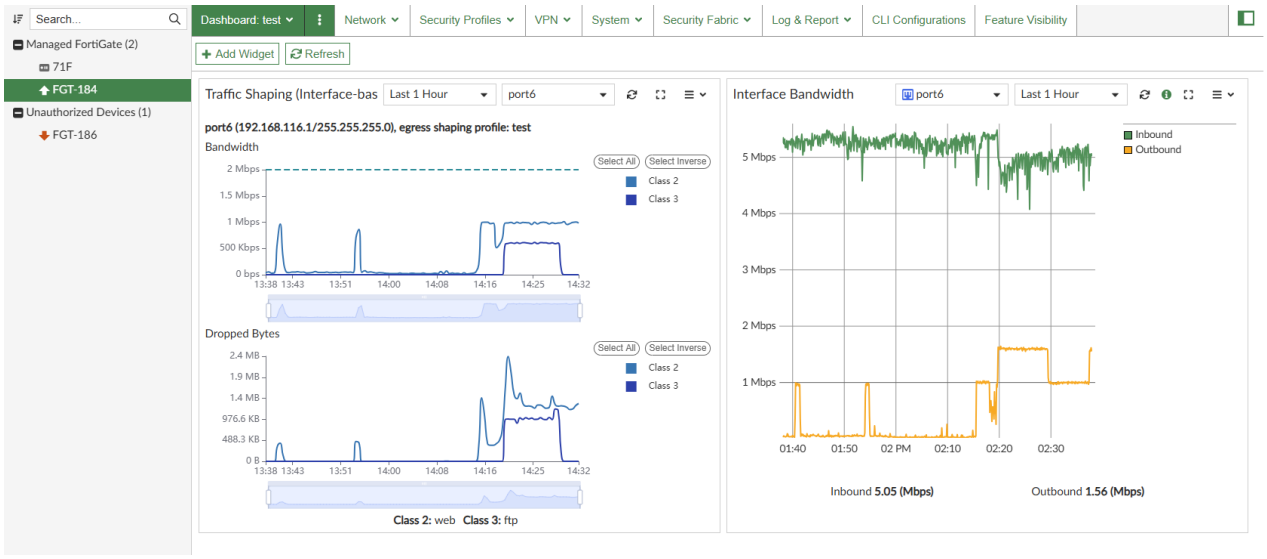
This method uses the same averaging method as the *Interface Bandwidth* widget, and aligns it with the visualization method used in the *Interface Bandwidth* widget.

Example

- Traffic Shaping graph using current bandwidth (legacy)



- Traffic Shaping graph using averaged bandwidth (new)


Search...
Managed FortiGate (2)
71F
FGT-184
Unauthorized Devices (1)
FGT-186
Dashboard: test
Network
Security Profiles
VPN
System
Security Fabric
Log & Report
CLI Configurations
Feature Visibility
Add Widget
Refresh
Traffic Shaping (Interface-bas
Last 1 Hour
port6
port6 (192.168.116.1/255.255.255.0), egress shaping profile: test
Bandwidth
2 Mbps
1.5 Mbps
1 Mbps
500 Kbps
0 bps
13:38 13:43 13:51 14:00 14:08 14:16 14:25 14:32
Select All
Select Inverse
Class 2
Class 3
Dropped Bytes
2.4 MB
1.9 MB
1.4 MB
976.6 KB
488.3 KB
0 B
Class 2: web Class 3: ftp
Interface Bandwidth
port6
Last 1 Hour
Inbound
Outbound
5 Mbps
4 Mbps
3 Mbps
2 Mbps
1 Mbps
01:40 01:50 02 PM 02:10 02:20 02:30
Inbound 5.05 (Mbps)
Outbound 1.56 (Mbps)

SD-WAN steering

8.0.0

SD-WAN underlay bandwidth steers traffic

A new option is available to allow load-balance service with hash-mode inbandwidth, outbandwidth, or bibandwidth to steer traffic based on shared underlay available bandwidth or individual overlay available bandwidth.

The `config service command` includes new options:

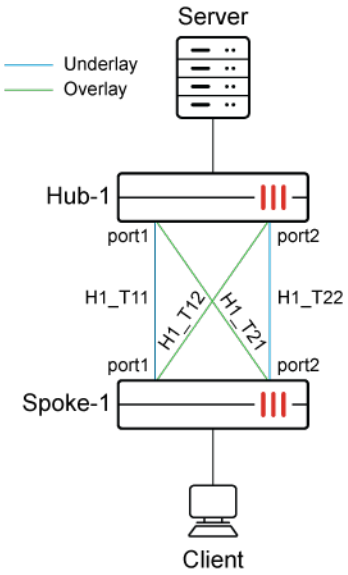
```
config system sdwan
    config service
        edit < num>
          set load-balance enable
          set hash-mode {round-robin | source-ip-based | source-dest-ip-based | inbandwidth |
outbandwidth | bibandwidth}
          set bandwidth-type {overlay* | underlay}
        next
    end
end
```

Option	Description
`hash-mode {round-robin | source-ip-based | source-dest-ip-based | inbandwidth | outbandwidth | bibandwidth}`	Hash algorithm for selected priority members for load-balance mode. Available when `load-balance` is enabled. Choose from: • `inbandwidth`: All traffic are distributed to a selected interface with most available bandwidth for incoming traffic. • `outbandwidth`: All traffic are distributed to a selected interface with most available bandwidth for outgoing traffic. • `bibandwidth`: All traffic are distributed to a selected interface with most available bandwidth for both incoming and outgoing traffic.
`bandwidth-type {overlay* | underlay}`	Overlay/underlay type of bandwidth (default = overlay).

Example

This example demonstrates how to use the underlay bandwidth to make traffic steering decisions for SD-WAN. In summary:

- Spoke-1 has four overlays: H1_T11, H1_T12, H1_T21, and H1_T22.
- H1_T11 and H1_T12 are associated with the same underlay port1, and H1_T21 and H1_T22 are associated with the same underlay port2.
- The load-balance service with `hash-mode bibandwidth` is configured on Spoke-1 to steer traffic on all four overlays.
- The overlay that is associated with the underlay with most available bandwidth is intended to carry traffic.



To configure Spoke-1:

1. Set the estimated bandwidth for upstream and downstream direction on each underlay and overlay:

```
config system interface
    edit "port1"
        ...
        set estimated-upstream-bandwidth 100000
        set estimated-downstream-bandwidth 100000
        ...
    next
    edit "port2"
        ...
        set estimated-upstream-bandwidth 50000
        set estimated-downstream-bandwidth 50000
        ...
    next
    edit "H1_T11"
        ...
        set estimated-upstream-bandwidth 90000
```

```
        set estimated-downstream-bandwidth 90000
         ...
    next
    edit "H1_T12"

        set estimated-upstream-bandwidth 90000
        set estimated-downstream-bandwidth 90000

    next
    edit "H1_T21"
        ...
        set estimated-upstream-bandwidth 40000
        set estimated-downstream-bandwidth 40000

    next
    edit "H1_T22"
        ...
        set estimated-upstream-bandwidth 40000
        set estimated-downstream-bandwidth 40000
        ...
    next
end
```

2. Configure SD-WAN with load-balancing enabled and `hash-mode` set to `bibandwidth`:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
        edit "overlay"
        next
        edit "underlay"
        next
    end
    config members
        edit 1
            set interface "port1"
            set zone "underlay"
        next
        edit 2
            set interface "port2"
            set zone "underlay"
            set gateway 172.31.3.6
        next
        edit 3
            set interface "H1_T11"
            set zone "overlay"
            set source 172.31.0.65
        next
        edit 4
            set interface "H1_T12"
```

```
            set zone "overlay"
            set source 172.31.0.65
        next
        edit 5
            set interface "H1_T21"
            set zone "overlay"
            set source 172.31.0.65
        next
        edit 6
            set interface "H1_T22"
            set zone "overlay"
            set source 172.31.0.65
        next
    end
    config health-check
        edit "HUB"
            set server "8.8.4.4"
            set update-static-route disable
            set members 3 4 5 6
            config sla
                edit 1
                    set link-cost-factor latency
                    set latency-threshold 100
                next
            end
        next
    end
    config service
        edit 1
            set name "1"
            set load-balance enable
            set mode sla
            set hash-mode bibandwidth
            set dst "all"
            set src "CORP_LAN"
            config sla
                edit "HUB"
                    set id 1
                next
            end
            set priority-members 3 4 5 6
        next
    end
end
```

3. By default, the service references the overlay available bandwidth:

```
Branch1_A_FGT (root) (Interim)# diagnose sys sdwan  service4

Service(1): Address Mode(IPV4) flags=0x24200 use-shortcut-sla use-shortcut load-balance
 Tie break: cfg
 Shortcut priority: 2
```

```
  Gen(1), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla hash-
mode=bibandwidth)
  Members(4):
    1: Seq_num(3 H1_T11 overlay), alive, sla(0x1), gid(2), num of pass(1), bibandwidth:
179995Kbps, selected                          // 90000 + 90000 = 180000
    2: Seq_num(4 H1_T12 overlay), alive, sla(0x1), gid(2), num of pass(1), bibandwidth:
179992Kbps, selected                          // 90000 + 90000 = 180000
    3: Seq_num(5 H1_T21 overlay), alive, sla(0x1), gid(2), num of pass(1), bibandwidth:
79997Kbps, selected                           // 40000 + 40000 = 80000
    4: Seq_num(6 H1_T22 overlay), alive, sla(0x1), gid(2), num of pass(1), bibandwidth:
79997Kbps, selected                           // 40000 + 40000 = 80000
  Src address(1):
        10.0.0.0-10.255.255.255
  Dst address(1):
        0.0.0.0-255.255.255.255
```

4. Configure underlay as `bandwidth-type` to have the service refer to underlay available bandwidth:

```
config sys sdwan
    config service
        edit 1
            set bandwidth-type underlay
        next
    end
end
```

5. Check the service.
   H1_T11/H1_12 has more available bibandwidth. H1_T11 is configured before H1_12, so H1_T11 is preferred.

```
Branch1_A_FGT (root) (Interim)# diagnose sys sdwan  service4

Service(1): Address Mode(IPV4) flags=0xa4200 use-shortcut-sla use-shortcut load-balance
 Tie break: cfg
 Shortcut priority: 2
  Gen(1), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla hash-
mode=bibandwidth)
  Members(4):
    1: Seq_num(3 H1_T11 overlay), alive, sla(0x1), gid(2), num of pass(1), bibandwidth:
199979Kbps, selected         // 100000 + 100000 = 200000
    2: Seq_num(4 H1_T12 overlay), alive, sla(0x1), gid(2), num of pass(1), bibandwidth:
199979Kbps, selected         // 100000 + 100000 = 200000
    3: Seq_num(5 H1_T21 overlay), alive, sla(0x1), gid(2), num of pass(1), bibandwidth:
99990Kbps, selected           // 50000 + 50000 = 100000
    4: Seq_num(6 H1_T22 overlay), alive, sla(0x1), gid(2), num of pass(1), bibandwidth:
99990Kbps, selected           // 50000 + 50000 = 100000
  Src address(1):
        10.0.0.0-10.255.255.255
  Dst address(1):
        0.0.0.0-255.255.255.255
```

6. Send around 83M traffic from the client to server, and check the SD-WAN status.
   83M traffic goes through H1_T11:

```
Branch1_A_FGT (root) (Interim)# diagnose sys sdwan intf-sla-log H1_T11

Timestamp: Wed Sep 24 11:34:44 2025, ifn: H1_T11, used inbandwidth: 10548bps, used
outbandwidth: 82640376bps, used bibandwidth: 82650924bps, tx bytes: 28265816236bytes, rx
bytes: 10078621bytes.
Timestamp: Wed Sep 24 11:34:54 2025, ifn: H1_T11, used inbandwidth: 7029bps, used
outbandwidth: 82321898bps, used bibandwidth: 82328927bps, tx bytes: 28367910559bytes, rx
bytes: 10078621bytes.


Branch1_A_FGT (root) (Interim)# diagnose sys sdwan intf-sla-log H1_T12

Timestamp: Wed Sep 24 11:36:35 2025, ifn: H1_T12, used inbandwidth: 2333bps, used
outbandwidth: 5032bps, used bibandwidth: 7365bps, tx bytes: 2862896972bytes, rx bytes:
6317363bytes.
Timestamp: Wed Sep 24 11:36:45 2025, ifn: H1_T12, used inbandwidth: 3464bps, used
outbandwidth: 5331bps, used bibandwidth: 8795bps, tx bytes: 2862903397bytes, rx bytes:
6324431bytes.

Branch1_A_FGT (root) (Interim)# diagnose sys sdwan intf-sla-log H1_T21

Timestamp: Wed Sep 24 11:39:05 2025, ifn: H1_T21, used inbandwidth: 2392bps, used
outbandwidth: 1468bps, used bibandwidth: 3860bps, tx bytes: 35828418453bytes, rx bytes:
4425608bytes.
Timestamp: Wed Sep 24 11:39:15 2025, ifn: H1_T21, used inbandwidth: 1580bps, used
outbandwidth: 1340bps, used bibandwidth: 2920bps, tx bytes: 35828419813bytes, rx bytes:
4425608bytes.

Branch1_A_FGT (root) (Interim)# diagnose sys sdwan intf-sla-log H1_T22

Timestamp: Wed Sep 24 11:39:35 2025, ifn: H1_T22, used inbandwidth: 1580bps, used
outbandwidth: 778bps, used bibandwidth: 2358bps, tx bytes: 54580737167bytes, rx bytes:
7183126bytes.
Timestamp: Wed Sep 24 11:39:45 2025, ifn: H1_T22, used inbandwidth: 2371bps, used
outbandwidth: 869bps, used bibandwidth: 3240bps, tx bytes: 54580738499bytes, rx bytes:
7188066bytes.
```

83M traffic goes through H1_T11's underlay:

```
Branch1_A_FGT (root) (Interim)# diagnose sys sdwan intf-sla-log port1

Timestamp: Wed Sep 24 11:40:35 2025, ifn: port1, used inbandwidth: 12103bps, used
outbandwidth: 86314630bps, used bibandwidth: 86326733bps, tx bytes: 36450897420bytes, rx
bytes: 18861999bytes.
Timestamp: Wed Sep 24 11:40:45 2025, ifn: port1, used inbandwidth: 12126bps, used
outbandwidth: 86321860bps, used bibandwidth: 86333986bps, tx bytes: 36558788532bytes, rx
bytes: 18877396bytes.

Branch1_A_FGT (root) (Interim)# diagnose sys sdwan intf-sla-log port2

Timestamp: Wed Sep 24 11:41:46 2025, ifn: port2, used inbandwidth: 4652bps, used outbandwidth:
5278bps, used bibandwidth: 9930bps, tx bytes: 94847045244bytes, rx bytes: 18330597bytes.
```

```
Timestamp: Wed Sep 24 11:41:56 2025, ifn: port2, used inbandwidth: 4728bps, used outbandwidth:
5375bps, used bibandwidth: 10103bps, tx bytes: 94847052353bytes, rx bytes: 18336819bytes.
```

For H1_T11, the underlay available bandwidth (roughly 200000 - 86333.986) is referenced. Although traffic is not on H1_T12, its underlay available bandwidth (roughly 200000 - 86333.986) is referenced as well.

```
Branch1_A_FGT (root) (Interim)# diagnose sys sdwan service4

Service(1): Address Mode(IPV4) flags=0xa4200 use-shortcut-sla use-shortcut load-balance
 Tie break: cfg
 Shortcut priority: 2
  Gen(1), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla hash-
mode=bibandwidth)
  Members(4):
    1: Seq_num(3 H1_T11 overlay), alive, sla(0x1), gid(2), num of pass(1), bibandwidth:
113673Kbps, selected
    2: Seq_num(4 H1_T12 overlay), alive, sla(0x1), gid(2), num of pass(1), bibandwidth:
113673Kbps, selected
    3: Seq_num(5 H1_T21 overlay), alive, sla(0x1), gid(2), num of pass(1), bibandwidth:
99990Kbps, selected
    4: Seq_num(6 H1_T22 overlay), alive, sla(0x1), gid(2), num of pass(1), bibandwidth:
99990Kbps, selected
  Src address(1):
        10.0.0.0-10.255.255.255
  Dst address(1):
        0.0.0.0-255.255.255.255.
```

7. Make H1_T11 out-of-sla.
   H1_T12 will be preferred as expected because it has more available bibandwidth within all in-sla members:

```
Branch1_A_FGT (root) (Interim)# diagnose sys sdwan  service4

Service(1): Address Mode(IPV4) flags=0xa4200 use-shortcut-sla use-shortcut load-balance
 Tie break: cfg
 Shortcut priority: 2
  Gen(2), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla hash-
mode=bibandwidth)
  Members(4):
    1: Seq_num(4 H1_T12 overlay), alive, sla(0x1), gid(2), num of pass(1), bibandwidth:
113673Kbps, selected
    2: Seq_num(5 H1_T21 overlay), alive, sla(0x1), gid(2), num of pass(1), bibandwidth:
99990Kbps, selected
    3: Seq_num(6 H1_T22 overlay), alive, sla(0x1), gid(2), num of pass(1), bibandwidth:
99990Kbps, selected
    4: Seq_num(3 H1_T11 overlay), alive, sla(0x0), gid(1), num of pass(0), bibandwidth:
113673Kbps, selected
  Src address(1):
        10.0.0.0-10.255.255.255
  Dst address(1):
        0.0.0.0-255.255.255.255
```

Selective SD-WAN duplication

SD-WAN duplication now supports selective member targeting, offering greater flexibility and control. Previously, duplication applied to all members in one zone indiscriminately.

The `config duplication` command includes new options:

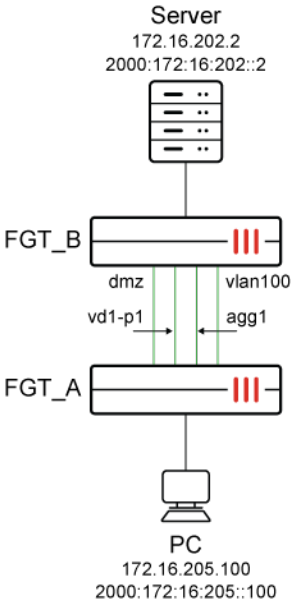
```
config system sdwan
    config duplication
      edit 1
        ...
        set members <*seq-num>
      next
    end
end
```

Option	Description
`members <*seq-num>`	Member sequence number list.

Example

This example demonstrates how to configure FGT_A of the SD-WAN topology to perform duplication on only two of four zone members.

FGT_A is configured with a zone (zone1) that allows a maximum of four zone members (dmz, vd1-p1, agg1, and vlan100) to perform duplication, and then duplication is further specified to occur only on members 1 and 2.



To configure duplication on FGT_A:

1. On FGT_A, enable SD-WAN, set a duplication maximum, create a zone with four members, set member priority, and define the duplication rule:

   In this example, member 1 is preferred for IPv4 and IPv6 traffic.

```
config system sdwan
    set status enable
    set duplication-max-num 4
    config zone
        edit "virtual-wan-link"
        next
        edit "zone1"
        next
    end
    config members
        edit 1
            set interface "dmz"
            set zone "zone1"
            set gateway 172.16.208.2
            set gateway6 2000:172:16:208::2
        next
        edit 2
            set interface "vd1-p1"
            set zone "zone1"
        next
        edit 3
            set interface "agg1"
            set zone "zone1"
            set gateway 172.16.203.2
            set gateway6 2000:172:16:203::2
        next
        edit 4
            set interface "vlan100"
            set zone "zone1"
            set gateway 172.16.206.2
            set gateway6 2000:172:16:206::2
        next
    end
    config service
        edit 1
            set name "1"
            set dst "all"
            set src "172.16.205.0"
            set priority-members 1 2 3 4
        next
        edit 2
            set name "2"
            set addr-mode ipv6
            set priority-members 1 2 3 4
            set dst6 "all"
            set src6 "2000:172:16:205::"
```

```
        next
    end
    config duplication
        edit 1
            set srcaddr "172.16.205.0"
            set dstaddr "172.16.202.0"
            set srcaddr6 "2000:172:16:205::"
            set dstaddr6 "2000:172:16:202::"
            set srcintf "port5"
            set dstintf "zone1"
            set service "PING" "PING6"
            set packet-duplication force
        next
    end
end
```

2. Configure IPv4 and IPv6 routes on members of zone1:

```
config router static
    ...
    edit 2
        set distance 1
        set sdwan-zone "virtual-wan-link" "zone1"
    next
end

config router static6
    ...
    edit 2
        set distance 1
        set sdwan-zone "virtual-wan-link" "zone1"
    next
end
```

3. Check IPv4 and IPv6 routing on which duplication relies:

```
get router info routing-table static
Routing table for VRF=0
S*      0.0.0.0/0 [1/0] via 172.16.203.2, agg1, [1/0]
                        [1/0] via 172.16.206.2, vlan100, [1/0]
                        [1/0] via 172.16.208.2, dmz, [1/0]
                        [1/0] via vd1-p1 tunnel 172.16.209.2, [1/0]

FGT_A (root) (Interim)# get router info6 routing-table static
Routing table for VRF=0
S*      ::/0 [1/0] via vd1-p1 tunnel ::172.16.209.2, 23:20:44, [1024/0]
                [1/0] via 2000:172:16:203::2, agg1, 23:20:44, [1024/0]
                [1/0] via 2000:172:16:206::2, vlan100, 23:20:44, [1024/0]
                [1/0] via 2000:172:16:208::2, dmz, 23:20:44, [1024/0]
```

4. Perform IPv4 and IPv6 ping from PC to server to see that packets are copied on all members in zone1:

```
~$ ping 172.16.202.2
PING 172.16.202.2 (172.16.202.2) 56(84) bytes of data.
64 bytes from 172.16.202.2: icmp_seq=1 ttl=253 time=0.577 ms
64 bytes from 172.16.202.2: icmp_seq=1 ttl=253 time=0.577 ms (DUP!)
64 bytes from 172.16.202.2: icmp_seq=1 ttl=253 time=0.577 ms (DUP!)
64 bytes from 172.16.202.2: icmp_seq=1 ttl=253 time=0.577 ms (DUP!)
64 bytes from 172.16.202.2: icmp_seq=2 ttl=253 time=0.467 ms
64 bytes from 172.16.202.2: icmp_seq=2 ttl=253 time=0.467 ms (DUP!)
64 bytes from 172.16.202.2: icmp_seq=2 ttl=253 time=0.467 ms (DUP!)
64 bytes from 172.16.202.2: icmp_seq=2 ttl=253 time=0.467 ms (DUP!)
^C
--- 172.16.202.2 ping statistics ---
2 packets transmitted, 2 received, +6 duplicates, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.467/0.522/0.577/0.055 ms

FGT_A (root) (Interim)# diagnose sniffer packet any 'host 172.16.202.2' 4
interfaces=[any]
filters=[host 172.16.202.2]
2.515725 port5 in 172.16.205.100 -> 172.16.202.2: icmp: echo request
2.515864 vd1-p1 out 172.16.205.100 -> 172.16.202.2: icmp: echo request           // member 2
2.515903 agg1 out 172.16.205.100 -> 172.16.202.2: icmp: echo request              // member 3
2.515909 port4 out 172.16.205.100 -> 172.16.202.2: icmp: echo request
2.515918 vlan100 out 172.16.205.100 -> 172.16.202.2: icmp: echo request         // member 4
2.515922 agg1 out 172.16.205.100 -> 172.16.202.2: icmp: echo request
2.515924 port4 out 172.16.205.100 -> 172.16.202.2: icmp: echo request
2.515930 dmz out 172.16.205.100 -> 172.16.202.2: icmp: echo request               // member 1
2.516292 vlan100 in 172.16.202.2 -> 172.16.205.100: icmp: echo reply
2.516347 vlan100 in 172.16.202.2 -> 172.16.205.100: icmp: echo reply
2.516362 vlan100 in 172.16.202.2 -> 172.16.205.100: icmp: echo reply
2.516375 vlan100 in 172.16.202.2 -> 172.16.205.100: icmp: echo reply
2.516431 port5 out 172.16.202.2 -> 172.16.205.100: icmp: echo reply
2.516437 port5 out 172.16.202.2 -> 172.16.205.100: icmp: echo reply
2.516441 port5 out 172.16.202.2 -> 172.16.205.100: icmp: echo reply
2.516444 port5 out 172.16.202.2 -> 172.16.205.100: icmp: echo reply


~$ ping 2000:172:16:202::2
PING 2000:172:16:202::2(2000:172:16:202::2) 56 data bytes
64 bytes from 2000:172:16:202::2: icmp_seq=1 ttl=62 time=0.766 ms
64 bytes from 2000:172:16:202::2: icmp_seq=1 ttl=62 time=0.766 ms (DUP!)
64 bytes from 2000:172:16:202::2: icmp_seq=1 ttl=62 time=0.766 ms (DUP!)
64 bytes from 2000:172:16:202::2: icmp_seq=1 ttl=62 time=0.766 ms (DUP!)
64 bytes from 2000:172:16:202::2: icmp_seq=2 ttl=62 time=0.539 ms
64 bytes from 2000:172:16:202::2: icmp_seq=2 ttl=62 time=0.539 ms (DUP!)
64 bytes from 2000:172:16:202::2: icmp_seq=2 ttl=62 time=0.539 ms (DUP!)
64 bytes from 2000:172:16:202::2: icmp_seq=2 ttl=62 time=0.539 ms (DUP!)
^C
--- 2000:172:16:202::2 ping statistics ---
2 packets transmitted, 2 received, +6 duplicates, 0% packet loss, time 1015ms
rtt min/avg/max/mdev = 0.539/0.652/0.766/0.116 ms

FGT_A (root) (Interim)#  diagnose sniffer packet any 'host 2000:172:16:202::2' 4
```

```
interfaces=[any]
filters=[host 2000:172:16:202::2]
3.400223 port5 in 2000:172:16:205::100 -> 2000:172:16:202::2: icmp6: echo request seq 1
[flowlabel 0xf62c8]
3.400370 vd1-p1 out 2000:172:16:205::100 -> 2000:172:16:202::2: icmp6: echo request seq 1
[flowlabel 0xf62c8]                    // member 2
3.400419 agg1 out 2000:172:16:205::100 -> 2000:172:16:202::2: icmp6: echo request seq 1
[flowlabel 0xf62c8]                       // member 3
3.400424 port4 out 2000:172:16:205::100 -> 2000:172:16:202::2: icmp6: echo request seq 1
[flowlabel 0xf62c8]
3.400435 vlan100 out 2000:172:16:205::100 -> 2000:172:16:202::2: icmp6: echo request seq 1
[flowlabel 0xf62c8]                  // member 4
3.400439 agg1 out 2000:172:16:205::100 -> 2000:172:16:202::2: icmp6: echo request seq 1
[flowlabel 0xf62c8]
3.400441 port4 out 2000:172:16:205::100 -> 2000:172:16:202::2: icmp6: echo request seq 1
[flowlabel 0xf62c8]
3.400449 dmz out 2000:172:16:205::100 -> 2000:172:16:202::2: icmp6: echo request seq 1
[flowlabel 0xf62c8]                        // member 1
3.400663 vlan100 in 2000:172:16:202::2 -> 2000:172:16:205::100: icmp6: echo reply seq 1
3.400764 vlan100 in 2000:172:16:202::2 -> 2000:172:16:205::100: icmp6: echo reply seq 1
3.400778 vlan100 in 2000:172:16:202::2 -> 2000:172:16:205::100: icmp6: echo reply seq 1
3.400791 vlan100 in 2000:172:16:202::2 -> 2000:172:16:205::100: icmp6: echo reply seq 1
3.400913 port5 out 2000:172:16:202::2 -> 2000:172:16:205::100: icmp6: echo reply seq 1
3.400921 port5 out 2000:172:16:202::2 -> 2000:172:16:205::100: icmp6: echo reply seq 1
3.400926 port5 out 2000:172:16:202::2 -> 2000:172:16:205::100: icmp6: echo reply seq 1
3.400930 port5 out 2000:172:16:202::2 -> 2000:172:16:205::100: icmp6: echo reply seq 1
```

5. Edit the service rule to specify zone members 1 and 2 for duplication:

```
config sys sdwan
    config duplication
        edit 1
            set member 1 2
    end
end
```

6. Perform IPv4 and IPv6 ping from PC to server to confirm packets are copied only on zone members 1 and 2:

```
~$ ping 172.16.202.2
PING 172.16.202.2 (172.16.202.2) 56(84) bytes of data.
64 bytes from 172.16.202.2: icmp_seq=1 ttl=253 time=0.762 ms
64 bytes from 172.16.202.2: icmp_seq=1 ttl=253 time=0.762 ms (DUP!)
64 bytes from 172.16.202.2: icmp_seq=2 ttl=253 time=0.471 ms
64 bytes from 172.16.202.2: icmp_seq=2 ttl=253 time=0.471 ms (DUP!)
^C
--- 172.16.202.2 ping statistics ---
2 packets transmitted, 2 received, +2 duplicates, 0% packet loss, time 1010ms
rtt min/avg/max/mdev = 0.471/0.616/0.762/0.147 ms


FGT_A (root) (Interim)#  diagnose sniffer packet any 'host 172.16.202.2' 4
interfaces=[any]
```

```
filters=[host 172.16.202.2]
3.740162 port5 in 172.16.205.100 -> 172.16.202.2: icmp: echo request
3.740262 vd1-p1 out 172.16.205.100 -> 172.16.202.2: icmp: echo request                    //
member 2
3.740299 dmz out 172.16.205.100 -> 172.16.202.2: icmp: echo request                        //
member 1
3.740566 vd1-p1 in 172.16.202.2 -> 172.16.205.100: icmp: echo reply
3.740591 port5 out 172.16.202.2 -> 172.16.205.100: icmp: echo reply
3.740596 vd1-p1 in 172.16.202.2 -> 172.16.205.100: icmp: echo reply
3.740607 port5 out 172.16.202.2 -> 172.16.205.100: icmp: echo reply


~$ ping 2000:172:16:202::2
PING 2000:172:16:202::2(2000:172:16:202::2) 56 data bytes
64 bytes from 2000:172:16:202::2: icmp_seq=1 ttl=62 time=0.730 ms
64 bytes from 2000:172:16:202::2: icmp_seq=1 ttl=62 time=0.754 ms (DUP!)
64 bytes from 2000:172:16:202::2: icmp_seq=2 ttl=62 time=0.473 ms
64 bytes from 2000:172:16:202::2: icmp_seq=2 ttl=62 time=0.473 ms (DUP!)
^C
--- 2000:172:16:202::2 ping statistics ---
2 packets transmitted, 2 received, +2 duplicates, 0% packet loss, time 1024ms
rtt min/avg/max/mdev = 0.473/0.607/0.754/0.137 ms


FGT_A (root) (Interim)#  diagnose sniffer packet any 'host 2000:172:16:202::2' 4
interfaces=[any]
filters=[host 2000:172:16:202::2]
2.233077 port5 in 2000:172:16:205::100 -> 2000:172:16:202::2: icmp6: echo request seq 1
[flowlabel 0xf62c8]
2.233210 vd1-p1 out 2000:172:16:205::100 -> 2000:172:16:202::2: icmp6: echo request seq 1
[flowlabel 0xf62c8]                  // member 2
2.233243 dmz out 2000:172:16:205::100 -> 2000:172:16:202::2: icmp6: echo request seq 1
[flowlabel 0xf62c8]                     // member 1
2.233542 vd1-p1 in 2000:172:16:202::2 -> 2000:172:16:205::100: icmp6: echo reply seq 1
2.233572 port5 out 2000:172:16:202::2 -> 2000:172:16:205::100: icmp6: echo reply seq 1
2.233579 vd1-p1 in 2000:172:16:202::2 -> 2000:172:16:205::100: icmp6: echo reply seq 1
2.233590 port5 out 2000:172:16:202::2 -> 2000:172:16:205::100: icmp6: echo reply seq 1
```

Control SD-WAN interface usage based on monthly traffic volume (quota)

Sometimes, SD-WAN underlay members may have monthly total bandwidth usage limits, which incurs overage costs with their ISPs once the limit is exceeded. To accomplish traffic steering based on monthly traffic volume, volume quota limit, billing start day, and related settings are added to SD-WAN members.

If accumulated traffic volume on one member exceeds the specified quota limit within one-month billing period, the member's cost, weight, or volume-ratio value can be automatically adjusted to force or redirect traffic to other members.

Traffic billing and steering based on monthly traffic volume supports high availability (HA).

The `config system sdwan` command has new options:

```
config system sdwan
    config members
      edit 1
        set quota-limit <integer>
        set billing-start-day <integer>
        set overage {enable | disable}
        set overage-cost <integer>
        set overage-weight <integer>
        set overage-volume-ratio <integer>
      next
    end
endd
```

Option	Description
`quota-limit <integer>`	Volume quota limit assigned to this member in gigabytes (0 - 10485760, default = 0).
`billing-start-day <integer>`	Volume billing start day when this member's volume usage will begin to calculate (1 - 31, default = 1). Available when `quota-limit` is set.
`overage {enable | disable}`	Enable/disable the volume overage when member's volume usage reaches quota-limit. Available when `quota-limit` is set.
`overage-cost <integer>`	Cost value for this member when its volume is over quota and overage is enabled (0 - 4294967295, default = 0). Available when `quota-limit` is set.
`overage-weight <integer>`	Weight value for this member when its volume is over quota and overage is enabled (0 - 255, default = 1). Available when `quota-limit` is set.
`overage-volume-ratio <integer>`	Volume ratio value for this member when its volume is over quota and overage is enabled (1 - 255, default = 1). Available when `quota-limit` is set.

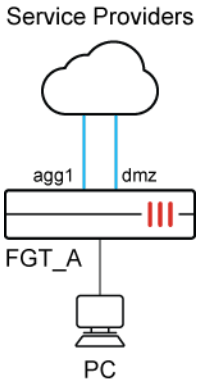
A new diagnose command is available:

```
# diagnose sys sdwan bill  member 1
```

Example

In this example,

- FGT_A has two SD-WAN members: agg1 and dmz.
- The service provider grants 2 GB bandwidth to agg1 per month. If 2 GB bandwidth on agg1 is reached within the one month billing period, traffic will switch to dmz for the rest of the billing period by increasing agg1's cost to deprioritize agg1.
- On the next billing day, 2 GB quota and cost will be reset and traffic switches back to agg1.



To configure interface usage based on monthly traffic volume:

1. Configure SD-WAN members.
   In this example:
   - Traffic volume limit is 2 GB within one-month billing period (`quota-limit 2`).
   - The start day is the first day in the month (`billing-start-day 1`).
   - The overage function is enabled (`overage enable`).
   - Cost 10 will be set when accumulated traffic volume on it exceeds 2 GB (`overage-cost 10`).
   - The overage weight will be set when accumulated traffic volume on it exceeds 2 GB (`overage-weight 1`).
   - The volume ratio will be set when accumulated traffic volume on it exceeds 2 GB (`overage-volume-ratio 1`).

   In addition, `mode sla` is set for IPv4 and IPv6 services.

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "agg1"
            set gateway 172.16.203.2
            set gateway6 2000:172:16:203::2
            set quota-limit 2
            set billing-start-day 1
            set overage enable
            set overage-cost 10
            set overage-weight 1
            set overage-volume-ratio 1
        next
```

```
        edit 2
            set interface "dmz"
            set gateway 172.16.208.2
            set gateway6 2000:172:16:208::2
        next
    end
    config health-check
        edit "1"
            set server "2.2.2.2"
            set update-static-route disable
            set members 0
            config sla
                edit 1
                next
            end
        next
        edit "2"
            set addr-mode ipv6
            set server "2000::2:2:2:2"
            set update-static-route disable
            set members 0
            config sla
                edit 1
                next
            end
        next
    end
    config service
        edit 1
            set name "1"
            set mode sla
            set dst "all"
            set src "172.16.205.0"
            config sla
                edit "1"
                    set id 1
                next
            end
            set priority-members 1 2
        next
        edit 2
            set name "2"
            set addr-mode ipv6
            set mode sla
            config sla
                edit "2"
                    set id 1
                next
            end
            set priority-members 1 2
            set dst6 "all"
```

```
            set src6 "2000:172:16:205::0"
        next
    end
end
```

2. Check SD-WAN status:
   The agg1 interface is preferred to forward IPv4 and IPv6 traffic. The 2 GB quota limit has not been reached.

```
# diagnose sys sdwan  service4

Service(1): Address Mode(IPV4) flags=0x4200 use-shortcut-sla use-shortcut
 Tie break: cfg
 Shortcut priority: 2
  Gen(1), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-compare-order
  Members(2):
    1: Seq_num(1 agg1 virtual-wan-link), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    2: Seq_num(2 dmz virtual-wan-link), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
  Src address(1):
        172.16.205.0-172.16.205.255
  Dst address(1):
        0.0.0.0-255.255.255.255

# diagnose sys sdwan  service6

Service(2): Address Mode(IPV6) flags=0x4200 use-shortcut-sla use-shortcut
 Tie break: cfg
 Shortcut priority: 2
  Gen(2), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-compare-order
  Members(2):
    1: Seq_num(1 agg1 virtual-wan-link), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    2: Seq_num(2 dmz virtual-wan-link), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
  Src6 address(1): 2000:172:16:205::/64
  Dst6 address(1): ::/0

# diagnose sys sdwan bill member 1
Member interface(agg1): Volume overage:enable, Quota_limit:2(GB), Volume is not overaged,
Reset_in_billing_day(0), Intf_set:up, Overage_cost(10), Overage_weight(1), Overage_volume(1),
Billing_start_day:1, Reset_accumulated_bytes:0, Current_bytes:234756493, Billing_start_
bytes:221780698, Billing_latest_bytes:234756049 Used_bytes since billing day:12975351
```

3. From the PC, initiate traffic.
   The traffic matches SD-WAN SLA-mode service and goes through agg1 on FGT_A for IPv4 and IPv6 traffic:

```
# diagnose sniffer packet any 'host 2.0.0.1' 4
interfaces=[any]
filters=[host 2.0.0.1]
4137.955796 port5 in 172.16.205.100 -> 2.0.0.1: icmp: echo request
4137.955895 agg1 out 172.16.205.100 -> 2.0.0.1: icmp: echo request
```

```
4137.955901 port4 out 172.16.205.100 -> 2.0.0.1: icmp: echo request
4137.956171 port4 in 2.0.0.1 -> 172.16.205.100: icmp: echo reply
4137.956176 agg1 in 2.0.0.1 -> 172.16.205.100: icmp: echo reply
4137.956203 port5 out 2.0.0.1 -> 172.16.205.100: icmp: echo reply


# diagnose sniffer packet any 'host 2000::2:0:0:1' 4
interfaces=[any]
filters=[host 2000::2:0:0:1]
4126.153276 port5 in 2000:172:16:205::100 -> 2000::2:0:0:1: icmp6: echo request seq 1
[flowlabel 0xb1863]
4126.153392 agg1 out 2000:172:16:205::100 -> 2000::2:0:0:1: icmp6: echo request seq 1
[flowlabel 0xb1863]
4126.153399 port4 out 2000:172:16:205::100 -> 2000::2:0:0:1: icmp6: echo request seq 1
[flowlabel 0xb1863]
4126.153577 port4 in 2000::2:0:0:1 -> 2000:172:16:205::100: icmp6: echo reply seq 1
4126.153585 agg1 in 2000::2:0:0:1 -> 2000:172:16:205::100: icmp6: echo reply seq 1
4126.153629 port5 out 2000::2:0:0:1 -> 2000:172:16:205::100: icmp6: echo reply seq 1
```

4. When accumulated traffic volume on agg1 exceeds 2 GB, traffic switches to dmz because the cost is increased to 10 in memory.

```
# diagnose sys sdwan bill  member 1
Member interface(agg1): Volume overage:enable, Quota_limit:2(GB), Volume is overaged, Reset_
in_billing_day(0) Intf_set:up, Overage_cost(10), Overage_weight(1), Overage_volume(1),
Billing_start_day:1,Reset_accumulated_bytes:0, Current_bytess:2231650140, Billing_start_
bytes:221780698, Billing_latest_bytes:2231647804 Used_bytes since billing day:2009867106

# diagnose sys sdwan  service4

Service(1): Address Mode(IPV4) flags=0x4200 use-shortcut-sla use-shortcut
 Tie break: cfg
 Shortcut priority: 2
  Gen(2), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-compare-order
  Members(2):
    1: Seq_num(2 dmz virtual-wan-link), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    2: Seq_num(1 agg1 virtual-wan-link), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
  Src address(1):
        172.16.205.0-172.16.205.255
  Dst address(1):
        0.0.0.0-255.255.255.255

# diagnose sys sdwan  service6

Service(2): Address Mode(IPV6) flags=0x4200 use-shortcut-sla use-shortcut
 Tie break: cfg
 Shortcut priority: 2
  Gen(2), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-compare-order
  Members(2):
    1: Seq_num(2 dmz virtual-wan-link), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
```

```
selected
    2: Seq_num(1 agg1 virtual-wan-link), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
  Src6 address(1): 2000:172:16:205::/64
  Dst6 address(1): ::/0

# diagnose sniffer packet any 'host 2.0.0.1' 4
interfaces=[any]
filters=[host 2.0.0.1]
....
6012.130521 port5 in 172.16.205.100 -> 2.0.0.1: icmp: echo request
6012.130612 dmz out 172.16.205.100 -> 2.0.0.1: icmp: echo request
6012.130784 dmz in 2.0.0.1 -> 172.16.205.100: icmp: echo reply
6012.130860 port5 out 2.0.0.1 -> 172.16.205.100: icmp: echo reply
6013.154504 port5 in 172.16.205.100 -> 2.0.0.1: icmp: echo request
6013.154557 dmz out 172.16.205.100 -> 2.0.0.1: icmp: echo request

# diagnose sniffer packet any 'host 2000::2:0:0:1' 4
interfaces=[any]
filters=[host 2000::2:0:0:1]
......
6002.927740 port5 in 2000:172:16:205::100 -> 2000::2:0:0:1: icmp6: echo request seq 1835
[flowlabel 0xb1863]
6002.927818 dmz out 2000:172:16:205::100 -> 2000::2:0:0:1: icmp6: echo request seq 1835
[flowlabel 0xb1863]
6002.927962 dmz in 2000::2:0:0:1 -> 2000:172:16:205::100: icmp6: echo reply seq 1835
6002.928009 port5 out 2000::2:0:0:1 -> 2000:172:16:205::100: icmp6: echo reply seq 1835
6003.947726 port5 in 2000:172:16:205::100 -> 2000::2:0:0:1: icmp6: echo request seq 1836
[flowlabel 0xb1863]
6003.947779 dmz out 2000:172:16:205::100 -> 2000::2:0:0:1: icmp6: echo request seq 1836
[flowlabel 0xb1863]
```

5. On the next billing day, 2 GB quota and cost are reset, and traffic switches back to agg1.

```
# diagnose sys sdwan bill  member 1
Member interface(agg1): Volume overage:enable, Quota_limit:2(GB), Volume is not overaged,
Reset_in_billing_day(1) Intf_set:up, Overage_cost(10), Overage_weight(1), Overage_volume(1),
Billing_start_day:1,Reset_accumulated_bytes:0, Current_bytess:2243437973, Billing_start_
bytes:2233333177, Billing_latest_bytes:2237462051 Used_bytes since billing day:4128874

# diagnose sys sdwan service4

Service(1): Address Mode(IPV4) flags=0x4200 use-shortcut-sla use-shortcut
 Tie break: cfg
 Shortcut priority: 2
  Gen(3), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-compare-order
  Members(2):
    1: Seq_num(1 agg1 virtual-wan-link), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    2: Seq_num(2 dmz virtual-wan-link), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
  Src address(1):
```

```
        172.16.205.0-172.16.205.255
  Dst address(1):
        0.0.0.0-255.255.255.255

# diagnose sys sdwan service6

Service(2): Address Mode(IPV6) flags=0x4200 use-shortcut-sla use-shortcut
 Tie break: cfg
 Shortcut priority: 2
  Gen(3), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-compare-order
  Members(2):
    1: Seq_num(1 agg1 virtual-wan-link), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    2: Seq_num(2 dmz virtual-wan-link), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
  Src6 address(1): 2000:172:16:205::/64
  Dst6 address(1): ::/0

# diagnose sniffer packet any 'host 2.0.0.1' 4
interfaces=[any]
filters=[host 2.0.0.1]

......

352023.096120 port5 in 172.16.205.100 -> 2.0.0.1: icmp: echo request
352023.096170 agg1 out 172.16.205.100 -> 2.0.0.1: icmp: echo request
352023.096175 port4 out 172.16.205.100 -> 2.0.0.1: icmp: echo request
352023.096325 port4 in 2.0.0.1 -> 172.16.205.100: icmp: echo reply
352023.096331 agg1 in 2.0.0.1 -> 172.16.205.100: icmp: echo reply
352023.096373 port5 out 2.0.0.1 -> 172.16.205.100: icmp: echo reply
352024.120110 port5 in 172.16.205.100 -> 2.0.0.1: icmp: echo request
352024.120149 agg1 out 172.16.205.100 -> 2.0.0.1: icmp: echo request
352024.120153 port4 out 172.16.205.100 -> 2.0.0.1: icmp: echo request

# diagnose sniffer packet any 'host 2000::2:0:0:1' 4
interfaces=[any]
filters=[host 2000::2:0:0:1]
.......
352013.825337 port5 in 2000:172:16:205::100 -> 2000::2:0:0:1: icmp6: echo request seq 2237
[flowlabel 0xb1863]
352013.825413 agg1 out 2000:172:16:205::100 -> 2000::2:0:0:1: icmp6: echo request seq 2237
[flowlabel 0xb1863]
352013.825418 port4 out 2000:172:16:205::100 -> 2000::2:0:0:1: icmp6: echo request seq 2237
[flowlabel 0xb1863]
352013.825594 port4 in 2000::2:0:0:1 -> 2000:172:16:205::100: icmp6: echo reply seq 2237
352013.825601 agg1 in 2000::2:0:0:1 -> 2000:172:16:205::100: icmp6: echo reply seq 2237
352013.825643 port5 out 2000::2:0:0:1 -> 2000:172:16:205::100: icmp6: echo reply seq 2237
352014.849326 port5 in 2000:172:16:205::100 -> 2000::2:0:0:1: icmp6: echo request seq 2238
[flowlabel 0xb1863]
352014.849377 agg1 out 2000:172:16:205::100 -> 2000::2:0:0:1: icmp6: echo request seq 2238
[flowlabel 0xb1863]
352014.849382 port4 out 2000:172:16:205::100 -> 2000::2:0:0:1: icmp6: echo request seq 2238
```

```
[flowlabel 0xb1863]
```

Service

Overlay as a Service

FortiCloud Overlay as a Service (OaaS) is a service for FortiGate devices to easily provision new SD-WAN overlay networks from FortiCloud. OaaS is a subscription service providing an easy-to-use GUI wizard that simplifies the process of configuring an SD-WAN overlay within a single region.

The OaaS hub acts as a bridge to allow overlay shortcuts to be formed between your spokes.

OaaS and the spokes rely on Fortinet Inc.'s Auto-Discovery VPN (ADVPN), which allows the central hub to dynamically inform spokes about a better path for traffic between two spokes. ADVPN shortcut tunnels, also known as shortcuts, are formed between spokes, such as between branches and the data center, or between branches themselves so that traffic does not need to pass through the hub.

OaaS can be accessed at https://overlay-as-a-service.forticloud.com/.

www.fortinet.com