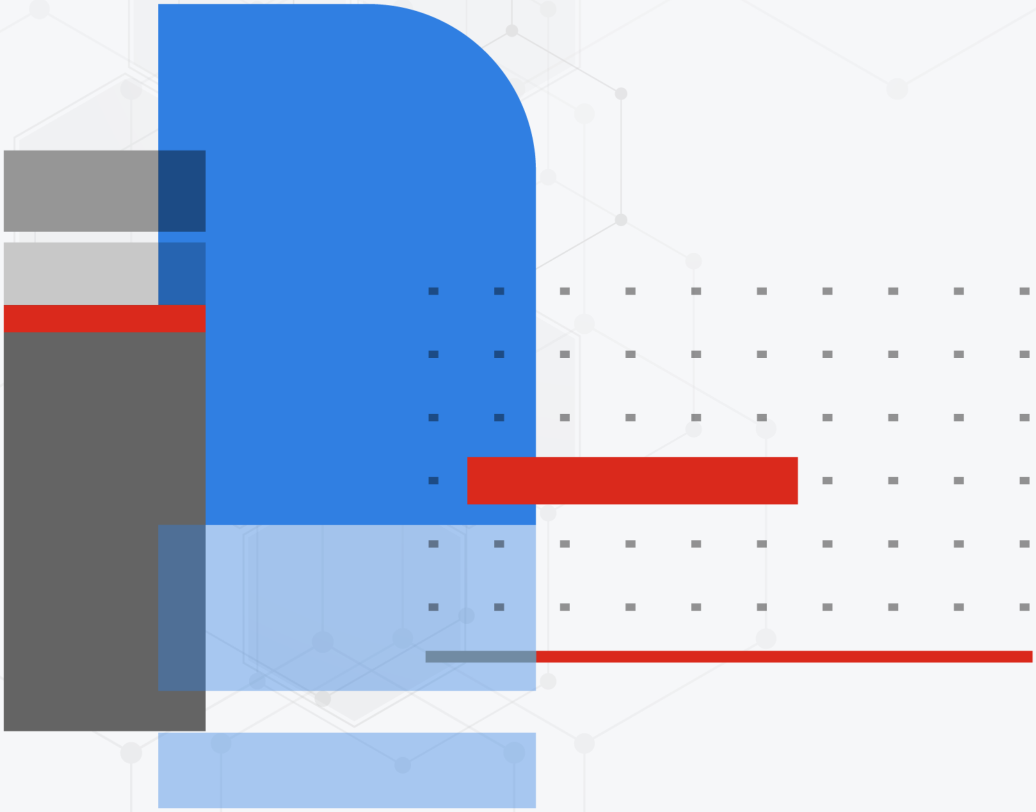
New Features Guide

SD-WAN with FortiOS, FortiManager, and FortiAnalyzer 7.4.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

December 20, 2023
SD-WAN with FortiOS, FortiManager, and FortiAnalyzer 7.4.x New Features Guide
01-74X-897981-20231220

TABLE OF CONTENTS

Change Log

Date	Change Description
2023-05-11	Initial release for FortiOS 7.4.0. See What's new in 7.4.0 on page 7.
2023-05-15	Updated to include FortiManager 7.4.0 features. See What's new in 7.4.0 on page 7.
2023-07-06	Updated Using MP-BGP EVPN with VXLAN on page 119.
2023-07-19	Updated Active SIM card switching available on FortiGates with cellular modem and dual SIM card support on page 16.
2023-08-31	Updated to include FortiOS and FortiManager 7.4.1 features. See What's new in 7.4.1 on page 8.
2023-09-07	Added Overlay as a Service on page 231.
2023-10-30	Added SD-WAN Cloud Assisted Monitoring service widgets FAZ 7.4.1 on page 73.
2023-12-20	Updated to include FortiOS and FortiManager 7.4.2 features. See What's new in 7.4.2 on page 8.

Overview

This guide provides details of new features for SD-WAN introduced in FortiOS 7.4, FortiManager 7.4, and FortiAnalyzer 7.4.

- What's new in 7.4.0 on page 7
- What's new in 7.4.1 on page 8
- What's new in 7.4.2 on page 8

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 FMG or FAZ: SD-WAN monitoring map enhancements FMG on page 51.

For features introduced in 7.4.1 and later versions, the version number is appended to the end of the topic heading. For example, Multiple interface monitoring for IPsec 7.4.1 on page 67 was introduced in 7.4.1. If a topic heading has no version number at the end, the feature was introduced in 7.4.0.

For features introduced in FortiManager or FortiAnalyzer 7.4.1 and later versions, the short product name and version number are appended to the end of the topic heading.

For example, Fabric Authorization Template is integrated with Device Blueprint and supports meta variables FMG 7.4.1 on page 97 was introduced in FortiManager 7.4.1.

What's new in 7.4.0

Feature	Details
ADVPN	• Add option to keep sessions in established ADVPN shortcuts while they remain in SLA on page 10 • Active SIM card switching available on FortiGates with cellular modem and dual SIM card support on page 16
Monitoring	• Logging FortiMonitor-detected performance metrics on page 45 • SD-WAN Monitoring Map integrates with Cloud Assisted Monitoring Service to allow FGT interface speed tests from inside FMG FMG on page 48 • SD-WAN monitoring map enhancements FMG on page 51
Provisioning	• Automated SD-WAN post overlay process creates policies to allow the health-check traffic to flow between Branch and HUB FMG on page 80 • Automated SD-WAN overlay process adds "branch_id" meta variable auto assignment FMG on page 83 • Fortinet factory-default wireless and extender templates FMG on page 85 • SD-WAN template for heterogeneous WAN link types FMG on page 92
Routing	• Using MP-BGP EVPN with VXLAN on page 119 • Add route tag address objects on page 130 • Allow better control over the source IP used by each egress interface for local

Feature	Details
	out traffic on page 132 - BGP conditional advertisements for IPv6 prefix when IPv4 prefix conditions are met and vice-versa on page 140
SD-WAN steering	- Classifying SLA probes for traffic prioritization on page 179 - Support IPv6 application based steering in SD-WAN on page 184 - Using a single IKE elector in ADVPN to match all SD-WAN control plane traffic on page 189 - VRF-aware SD-WAN IPv6 health checks on page 197 - Support maximize bandwidth (SLA) to load balance spoke-to-spoke traffic between multiple ADVPN shortcuts on page 198 - Allow multicast traffic to be steered by SD-WAN on page 205

What's new in 7.4.1

Feature	Details
ADVPN	- Active dynamic BGP neighbor triggered by ADVPN shortcut 7.4.1 on page 20
Monitoring	- Improve client-side settings for SD-WAN network monitor 7.4.1 on page 55 - Multiple interface monitoring for IPsec 7.4.1 on page 67 - SD-WAN Cloud Assisted Monitoring service widgets FAZ 7.4.1 on page 73
Provisioning	- Support the new SD-WAN Overlay-as-a-Service 7.4.1 on page 94 - Fabric Authorization Template is integrated with Device Blueprint and supports meta variables FMG 7.4.1 on page 97
Routing	- SD-WAN multi-PoP multi-hub large scale design and failover 7.4.1 on page 145
SD-WAN steering	- Support HTTPS performance SLA health checks 7.4.1 on page 219 - Using load balancing in a manual SD-WAN rule without configuring an SLA target 7.4.1 on page 221
Service	- Overlay as a Service on page 231

What's new in 7.4.2

Feature	Details
ADVPN	- ADVPN 2.0 edge discovery and path management 7.4.2 on page 31
Monitoring	- SD-WAN Monitoring dashboards allow full widget customization FMG 7.4.2 on page 77

Feature	Details
Provisioning	- Export Managed FortiAPs and import FortiAPs from a CSV file FMG 7.4.2 on page 100 - Meta variables are available in the SSID, FortiSwitch VLANs, and FortiSwitch Templates configuration FMG 7.4.2 on page 102 - FortiSwitch devices can be imported from a CSV file FMG 7.4.2 on page 107 - Factory default SSIDs and AP Profiles configuration updated FMG 7.4.2 on page 109
Routing	- Support IPsec tunnel to change names 7.4.2 on page 164 - Enhance IPv6 VRRP state control 7.4.2 on page 167 - SD-WAN hub and spoke speed test improvements 7.4.2 on page 169
SD-WAN steering	- IPv6 support for SD-WAN segmentation over a single overlay 7.4.2 on page 221 - BGP incorporates the advanced security measures of TCP Authentication Option (TCP-AO) 7.4.2 on page 228

ADVPN

7.4.0

- Add option to keep sessions in established ADVPN shortcuts while they remain in SLA on page 10
- Active SIM card switching available on FortiGates with cellular modem and dual SIM card support on page 16

7.4.1

- Active dynamic BGP neighbor triggered by ADVPN shortcut 7.4.1 on page 20

7.4.2

- ADVPN 2.0 edge discovery and path management 7.4.2 on page 31

Add option to keep sessions in established ADVPN shortcuts while they remain in SLA

This information is also available in the FortiOS 7.4 Administration Guide:

- Keeping sessions in established ADVPN shortcuts while they remain in SLA

In an SD-WAN hub and spoke configuration where ADVPN is used, when a primary shortcut goes out of SLA, traffic switches to the backup shortcut. During idle timeout, sessions will prefer using the primary parent tunnel and try to establish a new primary shortcut. However, because it is out of SLA, traffic switches back to the backup shortcut, which causes unnecessary traffic interruption.

The `shortcut-stickiness` option keeps existing sessions on the established ADVPN shortcuts while they remain in SLA instead of switching to a new link every idle timeout. New sessions will be routed through the primary shortcut if it is in SLA.

```
config system sdwan
    config service
        edit <id>
            set shortcut-stickiness {enable | disable}
        next
    end
end
```

The `shortcut-stickiness` option can be applied in the following use cases.

Use case 1:

1. The sessions will switch over to the backup shortcut due to the primary shortcut being out of SLA.
2. After an idle timeout, the primary shortcut is torn down, and the routes will be reinstalled on the primary parent tunnel.
3. When `shortcut-stickiness` is enabled, even though the primary parent tunnel is preferred, established ADVPN sessions will remain on the backup shortcut (stickiness) instead of switching to the primary parent tunnel.

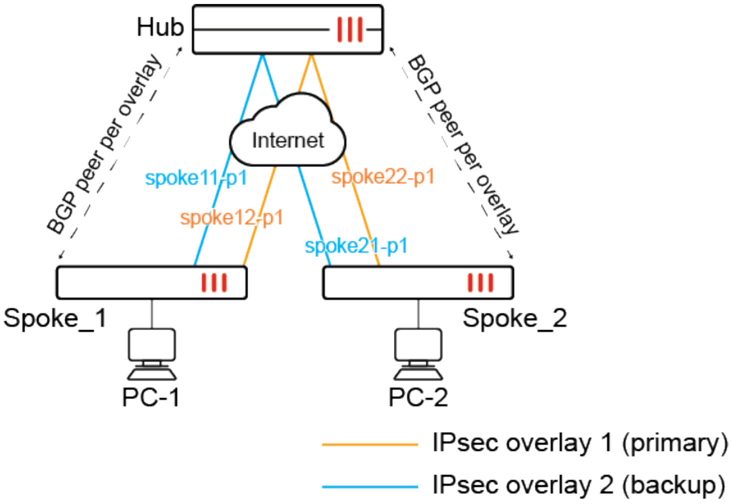
4. New sessions will be routed to the primary parent tunnel and trigger the primary shortcut, then traffic switches to the primary shortcut if it is in SLA.

Use case 2:

1. The sessions will switch over to the backup shortcut due to the primary shortcut being out of SLA.
2. After some time, the primary shortcut becomes in SLA.
3. When `shortcut-stickiness` is enabled, even though primary shortcut is preferred, established ADVPN sessions will remain on the backup shortcut (stickiness) instead of switching to the primary shortcut.
4. New sessions will be routed through the primary shortcut.

Example configuration

The following example demonstrates using the `shortcut-stickiness` option in use case 1.



After an idle timeout occurs, existing sessions remain on the spoke12-p1_0 backup shortcut tunnel. New sessions will try to create a shortcut over spoke11-p1, but will fall back to spoke12-p1_0 when it detects spoke11-p1 is out of SLA.

To configure shortcut stickiness for ADVPN shortcuts:

1. Configure SD-WAN on the Spoke_1 FortiGate:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "spoke11-p1"
        next
        edit 2
            set interface "spoke12-p1"
        next
    end
    config health-check
        edit "1"
            set server "9.0.0.1"
            set members 1 2
            config sla
                edit 1
                next
            end
        next
    end
    config service
        edit 1
            set name "1"
            set shortcut-stickiness enable
            set mode sla
            set dst "all"
            set src "10.1.100.0"
            config sla
                edit "1"
                    set id 1
                next
            end
            set priority-members 1 2
        next
    end
end
```

2. Verify the SD-WAN configuration.
    a. Verify the health check status:

```
# diagnose sys sdwan health-check
Health Check(1):
Seq(1 spoke11-p1): state(alive), packet-loss(0.000%) latency(0.368), jitter(0.051),
mos(4.404), bandwidth-up(999999), bandwidth-dw(1000000), bandwidth-bi(1999999) sla_
map=0x1
Seq(2 spoke12-p1): state(alive), packet-loss(0.000%) latency(0.211), jitter(0.019),
mos(4.404), bandwidth-up(999999), bandwidth-dw(999979), bandwidth-bi(1999978) sla_
map=0x1
```

    b. Verify the service status:

```
# diagnose sys sdwan service

Service(1): Address Mode(IPV4) flags=0x2200 use-shortcut-sla shortcut-stickiness
 Tie break: cfg
  Gen(1), TOS(0x0/0x0), Protocol(0: 1->65535), Mode(sla), sla-compare-order
  Members(2):
    1: Seq_num(1 spoke11-p1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    2: Seq_num(2 spoke12-p1), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
  Src address(1):
        10.1.100.0-10.1.100.255

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

The SD-WAN service rule prefers the primary parent tunnel (spoke11-p1) over the backup parent tunnel (spoke12-p1) before shortcuts are established.

3. Send traffic from PC-1 to PC-2 to trigger the primary shortcut. Verify the diagnostics.
   a. Run a sniffer trace:

```
# diagnose sniffer packet any 'host 192.168.5.44' 4
interfaces=[any]
filters=[host 192.168.5.44]
14.878761 port2 in 10.1.100.22 -> 192.168.5.44: icmp: echo request
14.878905 spoke11-p1 out 10.1.100.22 -> 192.168.5.44: icmp: echo request
14.879842 spoke11-p1 in 192.168.5.44 -> 10.1.100.22: icmp: echo reply
14.880082 port2 out 192.168.5.44 -> 10.1.100.22: icmp: echo reply
15.879761 port2 in 10.1.100.22 -> 192.168.5.44: icmp: echo request
15.879882 spoke11-p1_0 out 10.1.100.22 -> 192.168.5.44: icmp: echo request
15.880433 spoke11-p1_0 in 192.168.5.44 -> 10.1.100.22: icmp: echo reply
15.880496 port2 out 192.168.5.44 -> 10.1.100.22: icmp: echo reply
```

   The SD-WAN service rule sends traffic to the parent tunnel (spoke11-p1) initially, and then switches to the primary shortcut tunnel (spoke11-p1_0) once it is established.

   b. Verify the service status:

```
# diagnose sys sdwan service

Service(1): Address Mode(IPV4) flags=0x2200 use-shortcut-sla shortcut-stickiness
 Tie break: cfg
  Gen(2), TOS(0x0/0x0), Protocol(0: 1->65535), Mode(sla), sla-compare-order
  Member sub interface(3):
    2: seq_num(1), interface(spoke11-p1):
       1: spoke11-p1_0(57)
  Members(3):
    1: Seq_num(1 spoke11-p1_0), alive, sla(0x1), gid(0), cfg_order(0), local cost(0), 
selected
    2: Seq_num(1 spoke11-p1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0), 
selected
    3: Seq_num(2 spoke12-p1), alive, sla(0x1), gid(0), cfg_order(1), local cost(0), 
selected
  Src address(1):
        10.1.100.0-10.1.100.255

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

   The SD-WAN service rule prefers the primary shortcut tunnel (spoke11-p1_0) over other tunnels.

4. Make the primary shortcut be out of SLA. The traffic will switch to the backup parent tunnel and trigger the backup shortcut. Verify the diagnostics.
   a. Run a sniffer trace:

```
# diagnose sniffer packet any 'host 192.168.5.44' 4
interfaces=[any]
filters=[host 192.168.5.44]
20.588046 port2 in 10.1.100.22 -> 192.168.5.44: icmp: echo request
20.588157 spoke12-p1 out 10.1.100.22 -> 192.168.5.44: icmp: echo request
20.588791 spoke12-p1 in 192.168.5.44 -> 10.1.100.22: icmp: echo reply
20.588876 port2 out 192.168.5.44 -> 10.1.100.22: icmp: echo reply
21.589079 port2 in 10.1.100.22 -> 192.168.5.44: icmp: echo request
21.589190 spoke12-p1_0 out 10.1.100.22 -> 192.168.5.44: icmp: echo request
```

```
21.589661 spoke12-p1_0 in 192.168.5.44 -> 10.1.100.22: icmp: echo reply
21.589733 port2 out 192.168.5.44 -> 10.1.100.22: icmp: echo reply
```

When the primary shortcut tunnel goes out of SLA (spoke11-p1_0, alive, sla(0x0)), traffic reroutes to the backup parent tunnel (spoke12-p1) and then to the backup shortcut tunnel (spoke12-p1_0) once established.

b. Verify the service status:

```
# diagnose sys sdwan service

Service(1): Address Mode(IPV4) flags=0x2200 use-shortcut-sla shortcut-stickiness
 Tie break: cfg
  Gen(23), TOS(0x0/0x0), Protocol(0: 1->65535), Mode(sla), sla-compare-order
  Member sub interface(4):
    1: seq_num(1), interface(spoke11-p1):
       1: spoke11-p1_0(62)
    3: seq_num(2), interface(spoke12-p1):
       1: spoke12-p1_0(63)
  Members(4):
    1: Seq_num(1 spoke11-p1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0), 
selected
    2: Seq_num(2 spoke12-p1_0), alive, sla(0x1), gid(0), cfg_order(1), local cost(0), 
selected
    3: Seq_num(2 spoke12-p1), alive, sla(0x1), gid(0), cfg_order(1), local cost(0), 
selected
    4: Seq_num(1 spoke11-p1_0), alive, sla(0x0), gid(0), cfg_order(0), local cost(0), 
selected
  Src address(1):
        10.1.100.0-10.1.100.255

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

The backup shortcut tunnel (spoke12-p1_0) is now preferred.

5. After an idle timeout, the primary shortcut is torn down. The primary parent tunnel is now preferred, but traffic is still kept on the backup shortcut due to `shortcut-stickiness` being enabled. Verify the diagnostics.

a. Verify the service status:

```
# diagnose sys sdwan service

Service(1): Address Mode(IPV4) flags=0x2200 use-shortcut-sla shortcut-stickiness
 Tie break: cfg
  Gen(24), TOS(0x0/0x0), Protocol(0: 1->65535), Mode(sla), sla-compare-order
  Member sub interface(3):
    3: seq_num(2), interface(spoke12-p1):
       1: spoke12-p1_0(63)
  Members(3):
    1: Seq_num(1 spoke11-p1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0), 
selected
    2: Seq_num(2 spoke12-p1_0), alive, sla(0x1), gid(0), cfg_order(1), local cost(0), 
selected
    3: Seq_num(2 spoke12-p1), alive, sla(0x1), gid(0), cfg_order(1), local cost(0), 
selected
  Src address(1):
        10.1.100.0-10.1.100.255
  Dst address(1):
        0.0.0.0-255.255.255.255
```

b. Run a sniffer trace:

```
# diagnose sniffer packet any 'host 192.168.5.44' 4
interfaces=[any]
filters=[host 192.168.5.44]
1.065143 port2 in 10.1.100.22 -> 192.168.5.44: icmp: echo request
1.065218 spoke12-p1_0 out 10.1.100.22 -> 192.168.5.44: icmp: echo request
1.065471 spoke12-p1_0 in 192.168.5.44 -> 10.1.100.22: icmp: echo reply
1.065508 port2 out 192.168.5.44 -> 10.1.100.22: icmp: echo reply
2.066155 port2 in 10.1.100.22 -> 192.168.5.44: icmp: echo request
2.066198 spoke12-p1_0 out 10.1.100.22 -> 192.168.5.44: icmp: echo request
2.066442 spoke12-p1_0 in 192.168.5.44 -> 10.1.100.22: icmp: echo reply
2.066480 port2 out 192.168.5.44 -> 10.1.100.22: icmp: echo reply
3.067201 port2 in 10.1.100.22 -> 192.168.5.44: icmp: echo request
3.067255 spoke12-p1_0 out 10.1.100.22 -> 192.168.5.44: icmp: echo request
3.067507 spoke12-p1_0 in 192.168.5.44 -> 10.1.100.22: icmp: echo reply
3.067544 port2 out 192.168.5.44 -> 10.1.100.22: icmp: echo reply
```

6. Send new traffic from PC1 to PC2. The traffic is routed to the primary parent tunnel and triggers the primary shortcut, then traffic will switch to the primary shortcut if it is in SLA. Verify the connection.

a. Run a sniffer trace:

```
# diagnose sniffer packet any 'host 192.168.5.4' 4
interfaces=[any]
filters=[host 192.168.5.4]
17.120310 port2 in 10.1.100.22 -> 192.168.5.4: icmp: echo request
17.120475 spoke11-p1 out 10.1.100.22 -> 192.168.5.4: icmp: echo request
17.121096 spoke11-p1 in 192.168.5.4 -> 10.1.100.22: icmp: echo reply
17.121151 port2 out 192.168.5.4 -> 10.1.100.22: icmp: echo reply
18.121331 port2 in 10.1.100.22 -> 192.168.5.4: icmp: echo request
18.121480 spoke11-p1_0 out 10.1.100.22 -> 192.168.5.4: icmp: echo request
18.121954 spoke11-p1_0 in 192.168.5.4 -> 10.1.100.22: icmp: echo reply
18.122007 port2 out 192.168.5.4 -> 10.1.100.22: icmp: echo reply
...
```

At first, traffic tries to go to the primary parent tunnel so that it can trigger the primary shortcut to establish. The primary shortcut (spoke11-p1_0) is in SLA and new traffic flows through it.

```
...
14.194066 port2 in 10.1.100.22 -> 192.168.5.4: icmp: echo request
14.194247 spoke12-p1_0 out 10.1.100.22 -> 192.168.5.4: icmp: echo request
14.194499 spoke12-p1_0 in 192.168.5.4 -> 10.1.100.22: icmp: echo reply
14.194565 port2 out 192.168.5.4 -> 10.1.100.22: icmp: echo reply
15.195093 port2 in 10.1.100.22 -> 192.168.5.4: icmp: echo request
15.195174 spoke12-p1_0 out 10.1.100.22 -> 192.168.5.4: icmp: echo request
15.195326 spoke12-p1_0 in 192.168.5.4 -> 10.1.100.22: icmp: echo reply
15.195361 port2 out 192.168.5.4 -> 10.1.100.22: icmp: echo reply
```

After the primary shortcut goes out of SLA, the traffic switches to the backup shortcut (spoke12-p1_0).

b. Verify the service status:

```
# diagnose sys sdwan service

Service(1): Address Mode(IPV4) flags=0x2200 use-shortcut-sla shortcut-stickiness
```

```
 Tie break: cfg
  Gen(36), TOS(0x0/0x0), Protocol(0: 1->65535), Mode(sla), sla-compare-order
  Member sub interface(4):
    1: seq_num(1), interface(spoke11-p1):
       1: spoke11-p1_0(67)
    3: seq_num(2), interface(spoke12-p1):
       1: spoke12-p1_0(66)
  Members(4):
    1: Seq_num(1 spoke11-p1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    2: Seq_num(2 spoke12-p1_0), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    3: Seq_num(2 spoke12-p1), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    4: Seq_num(1 spoke11-p1_0), alive, sla(0x0), gid(0), cfg_order(0), local cost(0),
selected
  Src address(1):
        10.1.100.0-10.1.100.255

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

New traffic switches back to the backup shortcut while the primary shortcut is still out of SLA.

Active SIM card switching available on FortiGates with cellular modem and dual SIM card support

This information is also available in the FortiOS 7.4 Administration Guide:

- Active SIM card switching

FortiGates with a cellular modem and dual SIM card can switch in real time from the active SIM card to the passive SIM card when any of the following issues arise with the active SIM card:

- Ping link monitor fails. The SIM switch time depends on the link monitor parameters set.
- An active SIM card cannot be detected. The SIM switch time is about 20 seconds after the SIM card is no longer detected.
- A modem disconnection is detected, and a specified interval has elapsed. The SIM switch time occurs after the specified interval.

SIM card switching events are captured in the FortiGate event log.

In most cases, SIM cards come with the wireless carrier's APN, which is automatically retrieved at the first connection of the LTE modem. For these cases, you can use SIM cards for different wireless carriers in SIM slot 1 and slot 2.

When one or both SIM cards require their APN settings to be configured on the FortiGate, then both SIM cards should be for the same wireless carrier because `config system lte-modem` currently only supports a single `set apn < apn >` setting.

The following command and options can be used to configure this feature:

```
config system lte-modem
    config sim-switch
        set by-sim-state {enable | disable}
        set by-connection-state {enable | disable}
        set by-link-monitor {enable | disable}
        set link-monitor <link-monitor-name>
        set sim-switch-log-alert-interval <interval>
        set sim-switch-log-alert-threshold <threshold>
        set modem-disconnection-time <integer>
    end
end
```

`by-sim-state {enable | disable}`	Enable switching based on active SIM card state: • `enable`: switch to the passive SIM card whenever FortiGate cannot detect the active SIM card, such as when the active SIM card is ejected. • `disable`: do not switch SIM cards based on state.
`by-connection-state {enable | disable}`	Enable switching based on the connection state of the active SIM card: • `enable`: switch to the passive SIM card whenever FortiGate detects a modem signal loss after the `modem-disconnection-time` expires. • `disable`: do not switch SIM cards based on the connection state.
`by-link-monitor {enable | disable}`	Enable switching when a configured link monitor fails: • `enable`: switch to the passive SIM card when a link monitor configured with `link-monitor-name` fails. • `disable`: do not switch SIM cards based on the failure of a configured link monitor.
`link-monitor <link-monitor-name>`	Specify the name of the link monitor to use with `by-link-monitor`.
`sim-switch-log-alert-interval <interval>`	Identify what number of constant SIM card switch events will trigger an event log after the threshold in `sim-switch-log-alert-threshold` is met.
`sim-switch-log-alert-threshold`	Specify how many minutes to wait before creating an event log when the number of SIM card switches defined in `sim-switch-log-alert-interval` is met.
`modem-disconnection-time <integer>`	Specify how many seconds to wait before switching over to the passive SIM card when `by-connection-state` is enabled and a modem signal loss is detected.

Example 1

In this example, automatic SIM card switching is disabled. When disabled, the SIM card only works in the default slot1, but you can manually switch the SIM card to slot2. Event logs include details about the SIM card switch.

To manually switch a SIM card:

1. Disable automatic SIM card switching:

```
config system lte-modem
    config sim-switch
        set by-sim-state disable
        set by-connection-state disable
        set by-link-monitor disable
        set sim-slot 1
    end
end
```

2. Manually switch the SIM card from slot1 to slot2, and run the following command:

   ```
   # execute lte-modem sim-switch
   ```

   The SIM card switch may take a few seconds. You can run `diagnose system lte-modem sim-info` to check the results.

   The following log is generated after unplugging an active SIM card:

   ```
   7: date=2023-05-02 time=10:41:05 eventtime=1683049264795418820 tz="-0700"
   logid="0100046518" type="event" subtype="system" level="information" vd="root"
   logdesc="LTE modem active SIM card switch event" msg="LTE modem active SIM card slot
   changed to 2 by user."
   ```

Example 2

In this section, automatic SIM card switching is enabled and configured to switch based on SIM state, connection state, or link monitor state, and it includes example event logs for each scenario.

To enable automatic SIM card switching by SIM state:

1. Enable automatic SIM card switching by SIM state:

   ```
   config system lte-modem
       config sim-switch
           set by-sim-state enable
       end
   end
   ```

   With this configuration, the second SIM card becomes active when the active SIM card is no longer detected, for example, if the active SIM card is ejected. The following event logs are generated:

   ```
   5: date=2023-04-28 time=17:27:27 eventtime=1682728046989682780 tz="-0700"
   logid="0100046513" type="event" subtype="system" level="information" vd="root"
   logdesc="LTE modem data link connection event" msg="LTE modem data link changed from
   QMI_WDS_CONNECTION_STATUS_DISCONNECTED to QMI_WDS_CONNECTION_STATUS_CONNECTED"

   6: date=2023-04-28 time=17:27:17 eventtime=1682728036493684280 tz="-0700"
   logid="0100046512" type="event" subtype="system" level="information" vd="root"
   logdesc="LTE modem SIM card state event" msg="LTE modem SIM card change from QMI_UIM_
   CARD_STATE_ABSENT to QMI_UIM_CARD_STATE_PRESENT"

   7: date=2023-04-28 time=17:27:12 eventtime=1682728032589776580 tz="-0700"
   logid="0100046513" type="event" subtype="system" level="information" vd="root"
   logdesc="LTE modem data link connection event" msg="LTE modem data link changed from
   QMI_WDS_CONNECTION_STATUS_CONNECTED to QMI_WDS_CONNECTION_STATUS_DISCONNECTED"

   8: date=2023-04-28 time=17:27:11 eventtime=1682728031245682560 tz="-0700"
   logid="0100046512" type="event" subtype="system" level="information" vd="root"
   logdesc="LTE modem SIM card state event" msg="LTE modem SIM card change from QMI_UIM_
   CARD_STATE_PRESENT to QMI_UIM_CARD_STATE_ABSENT"
   ```

To enable automatic SIM card switching by connection state:

1. Enable automatic SIM card switching by connection state:

```
config system lte-modem
    config sim-switch
        set by-connection-state enable
        set modem-disconnection-time 30
        set sim-switch-log-alert-interval 15
        set sim-switch-log-alert-threshold 5
    end
end
```

With this configuration, the second SIM card becomes active when the modem cannot establish a connection with the carrier through the active SIM card. For example, a FortiGate is in a room with poor signal quality. With this configuration, the SIM card switch is triggered after the modem is detected as disconnected for 30 seconds, and the following event log is generated:

```
56: date=2023-05-01 time=11:14:56 eventtime=1682964896356933480 tz="-0700"
logid="0100046519" type="event" subtype="system" level="notice" vd="root" logdesc="LTE
modem active SIM card switched: modem disconnection detected" msg="LTE modem active SIM
card slot changed to 2, due to modem connection down."

66: date=2023-05-01 time=11:14:13 eventtime=1682964852964869400 tz="-0700"
logid="0100046519" type="event" subtype="system" level="notice" vd="root" logdesc="LTE
modem active SIM card switched: modem disconnection detected" msg="LTE modem active SIM
card slot changed to 1, due to modem connection down."
```

When poor signal quality causes SIM cards to frequently switch back and forth, and the flapping rate occurs more than five times within the configured 15 minute time period, an event log is triggered to record the flapping severity:

```
65: date=2023-05-01 time=11:14:13 eventtime=1682964853083194400 tz="-0700"
logid="0100046521" type="event" subtype="system" level="warning" vd="root" logdesc="LTE
modem active SIM card slot flipped back and forth in short time" msg="LTE modem switched
SIM slot 8 times in last 15 minutes, which is greater than 5 times threshold."
```

To enable automatic SIM card switching based on link monitor:

1. Enable automatic SIM card switching by link monitor, and specify the link monitor:

```
config system lte-modem
    config sim-switch
        set by-link-monitor enable
        set link-monitor "modem"
        set sim-switch-log-alert-interval 15
        set sim-switch-log-alert-threshold 5
    end
    config system link-monitor
    edit "modem"
        set srcintf "wwan"
        set server "8.8.8.8"
        set interval 1000
        set probe-timeout 100
        set failtime 3
        set recoverytime 8
    next
end
```

With this configuration, the second SIM card becomes active when the link monitor detects the active SIM card exceeds the SLA.

2. Check the link monitor status. In this example, the link monitor status is dead:

```
# diagnose system link-monitor status modem

Link Monitor: modem, Status: dead, Server num(1), cfg_version=7 HA state: local(dead),
shared(dead)
Flags=0x9 init log_downgateway, Create time: Fri Apr 28 16:34:56 2023
Source interface: wwan (19)
VRF: 0
Interval: 1000 ms
Service-detect: disable
Diffservcode: 000000
Class-ID: 0
  Peer: 8.8.8.8(8.8.8.8)
        Source IP(10.192.195.164)
        Route: 10.192.195.164->8.8.8.8/32, gwy(10.192.195.165)
        protocol: ping, state: dead
                Packet lost: 11.667%
                MOS: 4.353
                Number of out-of-sequence packets: 0
                Recovery times(5/8) Fail Times(1/3)
                Packet sent: 60, received: 56, Sequence(sent/rcvd/exp): 61/61/62
```

The following event log is generated when the link-monitor status is dead:

```
15: date=2023-04-28 time=16:31:38 eventtime=1682724697936494139 tz="-0700"
logid="0100046520" type="event" subtype="system" level="notice" vd="root" logdesc="LTE
modem active SIM card switched: link monitor probe failure detected" msg="LTE modem
active SIM card slot changed to 2, due to link monitor probe failures."

19: date=2023-04-28 time=16:31:13 eventtime=1682724673152506599 tz="-0700"
logid="0100022932" type="event" subtype="system" level="warning" vd="root" logdesc="Link
monitor status warning" name="modem" interface="wwan" probeproto="ping" msg="Link
Monitor changed state from alive to dead, protocol: ping."
```

Active dynamic BGP neighbor triggered by ADVPN shortcut - 7.4.1

When a customer using SD-WAN with ADVPN has numerous IPv4 and IPv6 routes per spoke and there are many spokes in the topology, using ADVPN with a route reflector-based design poses the following challenges:

- The hub FortiGate will experience high CPU usage due to the amount of processing required to reflect the routes to the spoke FortiGates.
- Spoke FortiGates will learn many unnecessary routes.

For such cases, it is more suitable to deploy an IPv4- and IPv6-supported solution without a route-reflector that involves an active dynamic BGP neighbor triggered by an ADVPN shortcut. This solution allows a spoke FortiGate to form a BGP neighbor with another spoke FortiGate only after the shortcut tunnel between them has been established. As a result, the spoke only learns routes from its BGP neighbors.

How this solution differs from typical SD-WAN with ADVPN

In a topology where the Spoke 1 and Spoke 2 FortiGates are connected directly to the Hub FortiGate, route reflection will not be enabled. The Hub FortiGate is only configured with each spoke's summary route. An ADVPN shortcut tunnel is established between the Spoke 1 and Spoke 2 FortiGates. The valid routing between the Spoke 1 and Spoke 2 FortiGate is still through the Hub FortiGate at this point.

When a host behind Spoke 1 tries to connect to a host behind Spoke 2, Spoke 1 first reaches the Hub based on the valid routing table. The Hub determines that the destination is reachable, and the ADVPN shortcut tunnel between the spokes is established. Then, Spoke 1 and Spoke 2 will actively initiate a BGP connection to each other over the shortcut. Once established, they will exchange their routing information using BGP. On both spokes, BGP will resolve those routes on the shortcut and update the routing table accordingly.

For this solution, the following IPv4/IPv6 BGP configuration settings are required:

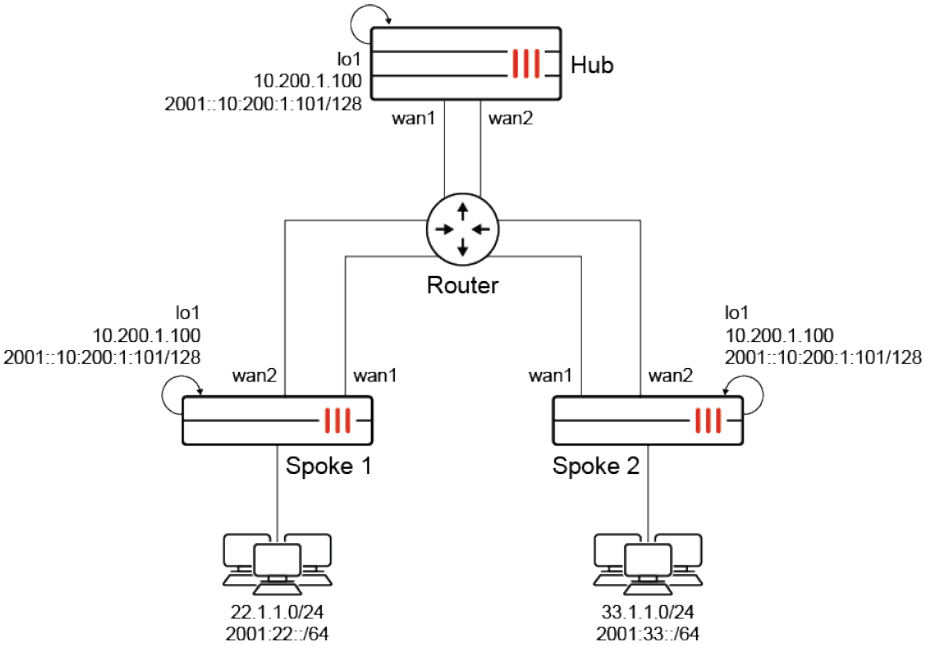
- The hub FortiGate should be configured with `neighbor-group` and `neighbor-range`/`neighbor-range6`.
- Each spoke FortiGate should be configured with `neighbor-group` and `neighbor-range`/`neighbor-range6` like the hub. More importantly, each spoke should be configured with `set passive disable` to ensure spokes are able to initiate dynamic BGP connections between each other.
- The hub FortiGate should have route reflection disabled (by default) where each `neighbor-group` setting should have `set route-reflector-client disable`.

In the configuration, each of the spokes will form a BGP neighbor relationship with the hub. This is unchanged from the typical SD-WAN with ADVPN configuration.

Example

This example configuration contains the following structure:

- Use SD-WAN member 1 (via ISP1) and its dynamic shortcuts for Financial Department traffic.
- Use SD-WAN member 2 (via ISP2) and its dynamic shortcuts for Engineering Department traffic.
- Internal subnets of Spoke 1:
  - IPv4: 22.1.1.0/24
  - IPv6: 2001:22::/64
- Internal subnets of Spoke 2:
  - IPv4: 33.1.1.0/24
    - Financial Department: 33.1.1.1 to 33.1.1.100
    - Engineering Department: 33.1.1.101 to 33.1.1.200
  - IPv6: 2001:33::/64
    - Financial Department: 2001:33::1 to 2001:33::100
    - Engineering Department: 2001:33::101 to 2001:33::200



To configure the Hub FortiGate:

1. Configure the BGP settings (neighbor group and ranges):

```
config router bgp
    set as 65100
    set router-id 10.200.1.1
    set ibgp-multipath enable
    config neighbor-group
        edit "EDGE"
            set activate6 disable
            set remote-as 65100
            set update-source "lo1"
            set route-reflector-client disable
        next
        edit "EDGEv6"
            set activate disable
            set remote-as 65100
            set update-source "lo1"
            set route-reflector-client disable
        next
    end
    config neighbor-range
        edit 2
            set prefix 10.200.1.0 255.255.255.0
            set neighbor-group "EDGE"
        next
    end
    config neighbor-range6
        edit 2
            set prefix6 2001::10:200:1:0/112
            set neighbor-group "EDGEv6"
        next
    end
    config network
        edit 2
            set prefix 10.200.1.0 255.255.255.0
        next
        edit 4
            set prefix 33.0.0.0 255.0.0.0
        next
        edit 5
            set prefix 22.0.0.0 255.0.0.0
        next
    end
    config network6
        edit 4
            set prefix6 2001:33::/32
        next
        edit 2
            set prefix6 2001:22::/32
        next
    end
end
```

2. Configure the static routes.
   a. For IPv4:

```
config router static
    edit 33
        set dst 33.0.0.0 255.0.0.0
        set blackhole enable
        set vrf 0
    next
    edit 22
        set dst 22.0.0.0 255.0.0.0
        set blackhole enable
        set vrf 0
    next
end
```

   b. For IPv6:

```
config router static6
    edit 33
        set dst 2001:33::/32
        set blackhole enable
        set vrf 0
    next
    edit 22
        set dst 2001:22::/32
        set blackhole enable
        set vrf 0
    next
end
```

The following IPv4 summary routes are advertised:

- 33.0.0.0/8
- 22.0.0.0/8

The following IPv6 summary routes are advertised:

- 2001:33::/32
- 2001:22::/32

Because route reflection has been disabled in this example, initially, Spoke 1 will not know the local subnet of Spoke 2, and Spoke 2 will not know the local subnet of Spoke 1. Therefore, for traffic routing, summary routes are configured on the hub as blackhole routes and then advertised to the spokes using BGP.

For example, for traffic from the local subnet of Spoke 2 destined for the local subnet of Spoke 1:

- For the IPv4 case, the summary route 22.0.0.0/8, which includes the local subnet of Spoke 1 (22.1.1.0/24), is advertised to Spoke 2. When Spoke 2 sends traffic destined for 22.1.1.0/24 to the Hub, the Hub forwards this traffic to Spoke 1 since they are BGP neighbors.
- For the IPv6 case, the summary route 2001:22::/32, which includes the local subnet of Spoke 1 (2001:22::/64), is advertised to Spoke 2. When Spoke 2 sends traffic destined for 2001:22::/64 to the Hub, the Hub forwards this traffic to Spoke 1 since they are BGP neighbors.

Although traffic from spoke-to-spoke goes through the hub first, it is expected that the spoke will eventually go through the shortcut tunnel.

To configure the Spoke 1 FortiGate:

1. Configure the SD-WAN settings:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "spoke1-1"
            set cost 10
        next
        edit 2
            set interface "spoke-2"
            set cost 20
        next
    end
    config health-check
        edit "ping"
            set server "11.11.11.11"
            set source 10.200.1.100
            set members 1 2
            config sla
                edit 1
                    set latency-threshold 200
                    set jitter-threshold 50
                next
            end
        next
    end
    config service
        edit 1
            set dst "financial-department"
            set priority-members 1
        next
        edit 2
            set dst "engineering-department"
            set priority-members 2
        next
        edit 61
            set addr-mode ipv6
            set priority-members 1
            set dst6 "financial-department-IPv6"
        next
        edit 62
            set addr-mode ipv6
            set priority-members 2
            set dst6 "engineering-department-IPv6"
        next
    end
end
```

2. Configure the BGP settings (neighbor group and ranges):

```
config router bgp
    set as 65100
    set router-id 10.200.1.100
    set ibgp-multipath enable
    config neighbor
        edit "10.200.1.1"
            set activate6 disable
            set remote-as 65100
            set connect-timer 10
            set update-source "lo1"
        next
        edit "2001::10:200:1:1"
            set advertisement-interval 1
            set activate disable
            set remote-as 65100
            set update-source "lo1"
        next
    end
    config neighbor-group
        edit "spokes"
            set activate6 disable
            set passive disable
            set remote-as 65100
            set update-source "lo1"
        next
        edit "spokesv6"
            set activate disable
            set passive disable
            set remote-as 65100
            set update-source "lo1"
        next
    end
    config neighbor-range
        edit 1
            set prefix 10.200.1.0 255.255.255.0
            set neighbor-group "spokes"
        next
    end
    config neighbor-range6
        edit 1
            set prefix6 2001::10:200:1:0/112
            set neighbor-group "spokesv6"
        next
    end
    config network
        edit 3
            set prefix 22.1.1.0 255.255.255.0
        next
    end
    config network6
        edit 1
            set prefix6 2001:22::/64
        next
    end
end
```

Verifying the configuration before a spoke-to-spoke shortcut VPN is established

IPv4 use case

To verify the status on Spoke 1:

1. Verify the BGP status:

```
# get router info bgp summary
VRF 0 BGP router identifier 10.200.1.100, local AS number 65100
BGP table version is 5
1 BGP AS-PATH entries
0 BGP community entries
Neighbor   V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.200.1.1 4      65100     222     225        3    0    0 00:15:14        3
Total number of neighbors 1
```

2. Verify the BGP routing table:

```
# get router info routing-table bgp
Routing table for VRF=0
B       11.11.11.11/32 [200/0] via 10.200.1.1 (recursive via spoke1-1 tunnel 11.1.1.11), 
00:15:19
                                              (recursive via spoke1-2 tunnel 
111.1.1.11), 00:15:19, [1/0]
B       22.0.0.0/8 [200/0] via 10.200.1.1 (recursive via spoke1-1 tunnel 11.1.1.11), 
00:15:19
                                          (recursive via spoke1-2 tunnel 111.1.1.11), 
00:15:19, [1/0]
B       33.0.0.0/8 [200/0] via 10.200.1.1 (recursive via spoke1-1 tunnel 11.1.1.11), 
00:15:19
                                          (recursive via spoke1-2 tunnel 111.1.1.11), 
00:15:19, [1/0]
```

IPv6 use case

To verify the status on Spoke 1:

1. Verify the BGP status:

```
# get router info6 bgp summary
VRF 0 BGP router identifier 10.200.1.100, local AS number 65100
BGP table version is 6
1 BGP AS-PATH entries
0 BGP community entries
Neighbor         V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2001::10:200:1:1 4      65100     223     224        4    0    0 00:15:21        3
Total number of neighbors 1
```

2. Verify the BGP routing table:

```
# get router info6 routing-table bgp
Routing table for VRF=0
B       2001::11:11:11:11/128 [200/0] via 2001::10:200:1:1 (recursive via spoke1-1
tunnel ::11.1.1.11), 00:15:29
                                                           (recursive via spoke1-2
tunnel ::111.1.1.11), 00:15:29, [1024/0]
B       2001:22::/32 [200/0] via 2001::10:200:1:1 (recursive via spoke1-1 tunnel
::11.1.1.11), 00:15:29
                                                  (recursive via spoke1-2 tunnel
::111.1.1.11), 00:15:29, [1024/0]
B       2001:33::/32 [200/0] via 2001::10:200:1:1 (recursive via spoke1-1 tunnel
::11.1.1.11), 00:15:29
                                                  (recursive via spoke1-2 tunnel
::111.1.1.11), 00:15:29, [1024/0]
```

Verifying the configuration after a single spoke-to-spoke shortcut VPN is established

IPv4 use case

To trigger a single spoke-to-spoke shortcut VPN, on host 22.1.1.22, ping the host 33.1.1.33 in the Financial Department. Because of the SD-WAN rule, use SD-WAN member 1 (via ISP1) and its dynamic shortcuts to reach hosts in the Financial Department.

To verify the status on Spoke 1:

1. Verify the BGP status:

```
# get router info bgp summary
VRF 0 BGP router identifier 10.200.1.100, local AS number 65100
BGP table version is 6
1 BGP AS-PATH entries
0 BGP community entries
Neighbor     V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.200.1.1   4      65100     252     254        3    0    0 00:17:22        3
10.200.1.101 4      65100       6       6        5    0    0 00:00:14        1
Total number of neighbors 2
```

Spoke 1 has as its BGP neighbors:

- Hub FortiGate at 10.200.1.1
- Spoke 2 FortiGate at 10.200.1.101

2. Verify the BGP routing table:

```
# get router info routing-table bgp
Routing table for VRF=0
B       11.11.11.11/32 [200/0] via 10.200.1.1 (recursive via spoke1-1 tunnel 11.1.1.11),
00:17:26
                                              (recursive via spoke1-2 tunnel
111.1.1.11), 00:17:26, [1/0]
B       22.0.0.0/8 [200/0] via 10.200.1.1 (recursive via spoke1-1 tunnel 11.1.1.11),
00:17:26
                                          (recursive via spoke1-2 tunnel 111.1.1.11),
00:17:26, [1/0]
B       33.0.0.0/8 [200/0] via 10.200.1.1 (recursive via spoke1-1 tunnel 11.1.1.11),
00:17:26
                                          (recursive via spoke1-2 tunnel 111.1.1.11),
00:17:26, [1/0]
B       33.1.1.0/24 [200/0] via 10.200.1.101 (recursive via spoke1-1_0 tunnel 13.1.1.3),
00:00:18, [1/0]
```

The remote route learned from Spoke 2 through the spoke1_1_0 tunnel and using BGP is 33.1.1.0/24.

IPv6 use case

To trigger a single spoke-to-spoke shortcut VPN over IPv6, on host 2001:22::22/64, ping the host 2001:33::33/64 in the Financial Department. Because of the SD-WAN rule, use SD-WAN member 1 (via ISP1) and its dynamic shortcuts to reach hosts in the Financial Department.

To verify the status on Spoke 1:

1. Verify the BGP status:

```
# get router info6 bgp summary
VRF 0 BGP router identifier 10.200.1.100, local AS number 65100
BGP table version is 7
1 BGP AS-PATH entries
0 BGP community entries
Neighbor           V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2001::10:200:1:1   4      65100     253     254        4    0    0 00:17:28        3
2001::10:200:1:101 4      65100       7       7        6    0    0 00:00:21        1
Total number of neighbors 2
```

Spoke 1 has as its BGP neighbors:

- Hub FortiGate at 2001::10:200:1:1
- Spoke 2 FortiGate at 2001::10:200:1:101

2. Verify the BGP routing table:

```
# get router info6 routing-table bgp
Routing table for VRF=0
B       2001::11:11:11:11/128 [200/0] via 2001::10:200:1:1 (recursive via spoke1-1
tunnel ::11.1.1.11), 00:17:30
                                                           (recursive via spoke1-2
tunnel ::111.1.1.11), 00:17:30, [1024/0]
B       2001:22::/32 [200/0] via 2001::10:200:1:1 (recursive via spoke1-1 tunnel
::11.1.1.11), 00:17:30
                                                  (recursive via spoke1-2 tunnel
::111.1.1.11), 00:17:30, [1024/0]
B       2001:33::/32 [200/0] via 2001::10:200:1:1 (recursive via spoke1-1 tunnel
::11.1.1.11), 00:17:30
                                                  (recursive via spoke1-2 tunnel
::111.1.1.11), 00:17:30, [1024/0]
B       2001:33::/64 [200/0] via 2001::10:200:1:101 (recursive via spoke1-1_0 tunnel
::13.1.1.3), 00:00:24, [1024/0]
```

The remote route learned from Spoke 2 through the spoke1-1_0 tunnel and using BGP is 2001:33::/64.

Verifying the configuration after a second spoke-to-spoke shortcut VPN is established

IPv4 use case

To trigger a second spoke-to-spoke shortcut VPN, on host 22.1.1.22, ping the host 33.1.1.133 in the Engineering Department. Because of the SD-WAN rule, use SD-WAN member 2 (via ISP2) and its dynamic shortcuts to reach hosts in the Engineering Department.

To verify the status on Spoke 1:

1. Verify the BGP status:

```
# get router info bgp summary
VRF 0 BGP router identifier 10.200.1.100, local AS number 65100
BGP table version is 6
1 BGP AS-PATH entries
0 BGP community entries
Neighbor     V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.200.1.1   4      65100     263     265         3    0    0 00:18:12        3
10.200.1.101 4      65100      17      17         5    0    0 00:01:04        1
Total number of neighbors
```

   Spoke 1 continues to have its BGP neighbors:

   - Hub FortiGate at 10.200.1.1
   - Spoke 2 FortiGate at 10.200.1.101

2. Verify the BGP routing table:

```
# get router info routing-table bgp
Routing table for VRF=0
B       11.11.11.11/32 [200/0] via 10.200.1.1 (recursive via spoke1-1 tunnel 11.1.1.11),
00:18:17
                                              (recursive via spoke1-2 tunnel
111.1.1.11), 00:18:17, [1/0]
B       22.0.0.0/8 [200/0] via 10.200.1.1 (recursive via spoke1-1 tunnel 11.1.1.11),
00:18:17
                                          (recursive via spoke1-2 tunnel 111.1.1.11),
00:18:17, [1/0]
B       33.0.0.0/8 [200/0] via 10.200.1.1 (recursive via spoke1-1 tunnel 11.1.1.11),
00:18:17
                                          (recursive via spoke1-2 tunnel 111.1.1.11),
00:18:17, [1/0]
B       33.1.1.0/24 [200/0] via 10.200.1.101 (recursive via spoke1-1_0 tunnel 13.1.1.3),
00:01:09
                                             (recursive via spoke1-2_0 tunnel
113.1.1.3), 00:01:09, [1/0]
```

The remote route learned from Spoke 2 through the spoke1-2_0 tunnel and using BGP is 33.1.1.0/24.

IPv6 use case

To trigger a second spoke-to-spoke shortcut VPN over IPv6, on host 2001:22::22/64, ping the host 2001:33::133/64 in the Engineering Department. Because of the SD-WAN rule, use SD-WAN member 2 (via ISP2) and its dynamic shortcuts to reach hosts in the Engineering Department.

To verify the status on Spoke 1:

1. Verify the BGP status:

```
# get router info6 bgp summary
VRF 0 BGP router identifier 10.200.1.100, local AS number 65100
BGP table version is 7
1 BGP AS-PATH entries
0 BGP community entries
Neighbor           V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2001::10:200:1:1   4      65100     264     265        4    0    0 00:18:18        3
2001::10:200:1:101 4      65100      19      19        6    0    0 00:01:11        1
Total number of neighbors 2
```

   Spoke 1 continues to have its BGP neighbors:

   - Hub FortiGate at 2001::10:200:1:1
   - Spoke 2 FortiGate at 2001::10:200:1:101

2. Verify the BGP routing table:

```
# get router info6 routing-table bgp
Routing table for VRF=0
B       2001::11:11:11:11/128 [200/0] via 2001::10:200:1:1 (recursive via spoke1-1
tunnel ::11.1.1.11), 00:18:20
                                                           (recursive via spoke1-2
tunnel ::111.1.1.11), 00:18:20, [1024/0]
B       2001:22::/32 [200/0] via 2001::10:200:1:1 (recursive via spoke1-1 tunnel
::11.1.1.11), 00:18:20
                                                  (recursive via spoke1-2 tunnel
::111.1.1.11), 00:18:20, [1024/0]
B       2001:33::/32 [200/0] via 2001::10:200:1:1 (recursive via spoke1-1 tunnel
::11.1.1.11), 00:18:20
                                                  (recursive via spoke1-2 tunnel
::111.1.1.11), 00:18:20, [1024/0]
B       2001:33::/64 [200/0] via 2001::10:200:1:101 (recursive via spoke1-1_0 tunnel
::13.1.1.3), 00:01:14
                                                    (recursive via spoke1-2_0 tunnel
::113.1.1.3), 00:01:14, [1024/0]
```

   The remote route learned from Spoke 2 through the spoke1-2_0 tunnel and using BGP is 2001:33::/64.

ADVPN 2.0 edge discovery and path management - 7.4.2

The SD-WAN with ADVPN solution has evolved to version 2.0 with major changes to ADVPN design and operation, including the introduction of edge discovery and path management for ADVPN spokes.

ADVPN 2.0 incorporates intelligence into the spokes to ensure shortcut tunnels (also known as shortcuts) are established using underlays available on both spokes and chosen based on matching certain link health criteria.

ADVPN 2.0 provides a more flexible SD-WAN solution than the original ADVPN to achieve resiliency against underlay outages or degraded underlay performance because it no longer depends on specific BGP routing designs or mechanisms, including route reflection, BGP next hop recursive resolution, BGP per overlay, and BGP on loopback.

ADVPN 2.0 only supports IPv4.

The topic includes the following sections:

- Overview on page 31
- Example on page 32

Overview

The overview covers the following information:

- How this solution differs from SD-WAN with previous ADVPN on page 31
- SD-WAN CLI configuration commands on page 32

How this solution differs from SD-WAN with previous ADVPN

With the previous version of ADVPN and SD-WAN, shortcut path selection relied entirely on the overlays between the spokes. The hub and overlays were used to exchange IKE shortcut messages, and policy routes were configured on the hub to ensure shortcuts were established on the same overlay. In addition, user traffic was needed to trigger the process of establishing shortcuts.

With the latest version of ADVPN and SD-WAN, shortcut path selection is achieved through edge discovery and path management functionality on the ADVPN spokes.

1. Edge discovery:
    - Expand IKE Shortcut-Reply message to allow the local spoke (spoke where user traffic is initiated) to obtain the remote spoke (destination spoke for user traffic) WAN link information, which includes IP address, transport group, link quality, link cost, and member configuration order.
    - After shortcut establishment, WAN link information can be exchanged on the shortcut regularly every 5 seconds through UDP traffic. The path management function on the local spoke is regularly updated to pick up changes to remote or local overlays and select the best shortcut path accordingly.
2. Path management:
    - The local spoke handles the remote spoke WAN link information, calculates the best shortcut path per SD-WAN service or rule, and then advises IKE to establish a shortcut using the selected path.

SD-WAN CLI configuration commands

The following SD-WAN CLI configuration commands are used to configure ADVPN 2.0 on the spokes:

```
config system sdwan
    config zone
        edit <zone-name>
            set advpn-select {enable | disable}
            set advpn-health-check <health-check name>
        next
    end
    config members
        edit <integer>
            set transport-group <integer>
        next
    end
    config service
        edit <integer>
            set shortcut-priority {enable | disable | auto}
        next
    end
end
```

`set advpn-select {enable	disable}`	Enable or disable SDWAN/ADVPN-2.0 (default=disabled).
`set advpn-health-check <health-check name>`	Specify the health check for the spoke whose info will be sent to the peer spoke.
`set transport-group <integer>`	Specify different group ID between (1 -255) to differentiate link-type, such as Internet, MPLS, LTE, Satellite.
`set shortcut-priority {enable	disable	auto}`	Enable or disable making ADVPN shortcut a high priority over overlay parent interfaces, if SLA mode or link cost factor mode conditions are met: • `enable`: enable a high priority of ADVPN shortcut for this service. • `disable`: disable a high priority of ADVPN shortcut for this service. • `auto`: automatically enable a high priority of ADVPN shortcut for this service if ADVPN2.0 is enabled.
`diagnose sys sdwan advpn-session`	Diagnostic command run on local spoke to view remote spoke WAN link information and path manager shortcut path selection.

As with the previous version of ADVPN, on the hub, you must enable ADVPN and configure firewall policies between spokes.

Example

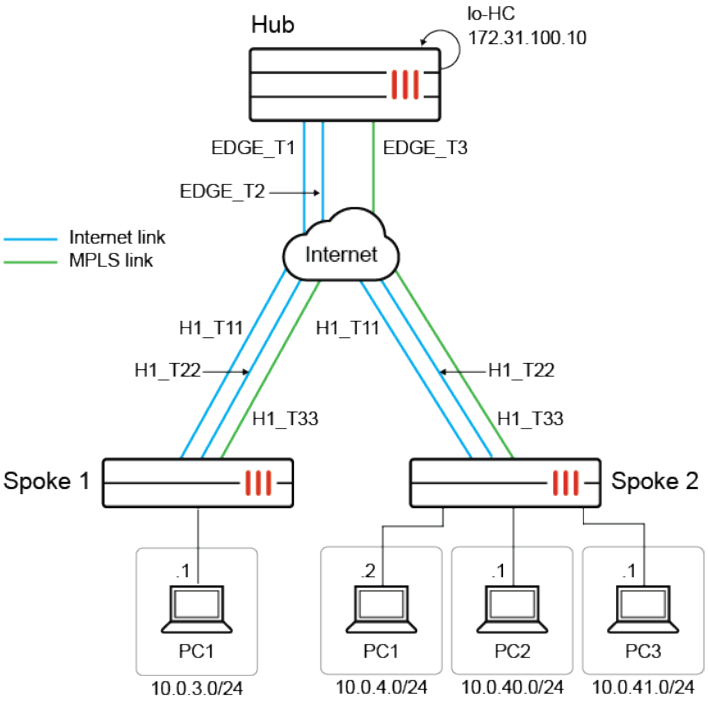
The configuration example illustrates the edge discovery and path management processes for a typical hub and spoke topology. This example focuses on SD-WAN configuration for steering traffic and establishing shortcuts in the direction from Spoke 1 to Spoke 2.

- Network Topology on page 33
- SD-WAN configuration and health check status on page 34
- Scenario 1: Traffic matching SD-WAN rule 1 on page 36

- Scenario 2: Traffic matching SD-WAN rule 2 on page 38
- Scenario 3: Traffic matching SD-WAN rule 3 on page 40
- Scenario 4: Spoke 2 H1_T22 overlay link out-of-SLA on page 41

Network Topology

In this example, BGP per overlay was used for dynamic routing to distribute the LAN routes behind each spoke to the other spoke. However, this was a design choice. You can also use BGP on loopback for this example.



Spokes 1 and 2 have the following VPN overlays between themselves and the hub:

VPN Overlays	IP address on Spoke 1	IP address on Spoke 2
H1_T11	172.31.80.1/32	172.31.80.2/32
H1_T22	172.31.81.1/32	172.31.81.2/32
H1_T33	172.31.82.1/32	172.31.82.2/32

SD-WAN Rules/Services defined on Spoke 1:

	SD-WAN Rule/Service 1	SD-WAN Rule/Service 2	SD-WAN Rule/Service 3
	H1_T11	H1_T22	H1_T33
	H1_T22	H1_T11	H1_T11
	H1_T33	H1_T33	H1_T22
Strategy for choosing outgoing interfaces	Lowest cost (SLA)	Lowest cost (SLA)	Best quality, link cost factor: packet loss

Throughout this example, transport group 1 is used for VPN overlays over Internet links while transport group 2 is used for the VPN overlay over an MPLS link.

In this example, user traffic is initiated behind Spoke 1 and destined to Spoke 2. Because of this, Spoke 1 is considered the local spoke, and Spoke 2 is considered the remote spoke.

SD-WAN configuration and health check status

SD-WAN configuration and health check status on Spoke 1:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
        edit "overlay"
            set advpn-select enable
            set advpn-health-check "HUB"
        next
    end
    config members
        edit 1
            set interface "H1_T11"
            set zone "overlay"
            set transport-group 1
        next
        edit 2
            set interface "H1_T22"
            set zone "overlay"
            set transport-group 1
        next
        edit 3
            set interface "H1_T33"
            set zone "overlay"
            set transport-group 2
        next
    end
    config health-check
        edit "HUB"
            set server "172.31.100.100"
            set members 1 2 3
            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 mode sla
            set shortcut-priority enable
            set dst "spoke-2_LAN-1" "Tunnel_IPs"
            set src "spoke-1_LAN-1" "Tunnel_IPs"
            config sla
                edit "HUB"
                    set id 1
                next
            end
            set priority-members 1 2 3
        next
        edit 2
            set name "2"
            set mode sla
            set shortcut-priority enable
            set dst "spoke-2_LAN-2" "Tunnel_IPs"
            set src "spoke-1_LAN-1" "Tunnel_IPs"
            config sla
                edit "HUB"
                    set id 1
                next
            end
            set priority-members 2 1 3
        next
        edit 3
            set name "3"
            set mode priority
            set dst "spoke-2_LAN-3" "Tunnel_IPs"
            set src "spoke-1_LAN-1" "Tunnel_IPs"
            set health-check "HUB"
            set link-cost-factor packet-loss
            set priority-members 3 1 2
        next
    end
end

# diagnose sys sdwan health-check
Health Check(HUB):
Seq(1 H1_T11): state(alive), packet-loss(0.000%) latency(0.231), jitter(0.029), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999997), bandwidth-bi(1999996) sla_map=0x1
Seq(2 H1_T22): state(alive), packet-loss(0.000%) latency(0.193), jitter(0.010), mos(4.404),
bandwidth-up(999994), bandwidth-dw(999997), bandwidth-bi(1999991) sla_map=0x1
Seq(3 H1_T33): state(alive), packet-loss(0.000%) latency(0.144), jitter(0.007), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999997), bandwidth-bi(1999996) sla_map=0x1
```

SD-WAN configuration and health check status on Spoke 2:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
        edit "overlay"
            set advpn-select enable
            set advpn-health-check "HUB"
        next
    end
    config members
```

```
        edit 1
            set interface "H1_T11"
            set zone "overlay"
            set cost 100
            set transport-group 1
        next
        edit 2
            set interface "H1_T22"
            set zone "overlay"
            set transport-group 1
        next
        edit 3
            set interface "H1_T33"
            set zone "overlay"
            set transport-group 2
        next
    end
    config health-check
        edit "HUB"
            set server "172.31.100.100"
            set members 3 1 2
            config sla
                edit 1
                    set link-cost-factor latency
                    set latency-threshold 100
                next
            end
        next
    end
end

# diagnose sys sdwan health-check
Health Check(HUB):
Seq(3 H1_T33): state(alive), packet-loss(0.000%) latency(0.124), jitter(0.009), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x1
Seq(1 H1_T11): state(alive), packet-loss(0.000%) latency(0.216), jitter(0.043), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x1
Seq(2 H1_T22): state(alive), packet-loss(0.000%) latency(0.184), jitter(0.012), mos(4.404),
bandwidth-up(999994), bandwidth-dw(999998), bandwidth-bi(1999992) sla_map=0x1
```

Scenario 1: Traffic matching SD-WAN rule 1

In this scenario, PC 1 connected to Spoke 1 initiates an ICMP ping destined for PC1 connected to Spoke 2. Therefore, this user traffic matches SD-WAN rule 1 and triggers shortcut path selection and establishment.

The Path Manager of Spoke 1 will calculate the best shortcut path by comparing transport group, link quality (for SLA mode), link cost, and member configuration order between Spoke 1 and Spoke 2.

For an SLA mode service, the following algorithm is followed for considering endpoints of the best shortcut path:

1. Overlays with the same transport group
2. In-SLA overlays
3. Lowest link cost overlays
4. Member configuration order as a final tiebreaker

Based on this algorithm, the Path Manager on Spoke 1 selects Spoke 1 H1_T11 because:

It is first in the priority-members order for SD-WAN rule 1, it has the lowest link cost, and it is within SLA. Likewise, the Path Manager on Spoke 1 selects Spoke 2 H1_T22 since it has the lowest link cost compared to Spoke 2 H1_T11 (which has a cost of 100), it is within SLA, and has the same transport group as Spoke 1 H1_T11. Therefore, the Path Manager of Spoke 1 calculates the best shortcut path as Spoke 1 H1_T11 to Spoke 2 H1_T22.

The Path Manager will advise IKE to establish the best shortcut and add it to SD-WAN rule 1 as follows:

```
Branch1_FGT# diagnose sys sdwan service
Service(1): Address Mode(IPV4) flags=0x4200 use-shortcut-sla use-shortcut
 Tie break: cfg
 Shortcut priority: 1
  Gen(11), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-compare-
order
  Member sub interface(4):
    2: seq_num(1), interface(H1_T11):
       1: H1_T11_0(71)
  Members(4):
    1: Seq_num(1 H1_T11_0 overlay), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    2: Seq_num(1 H1_T11 overlay), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    3: Seq_num(2 H1_T22 overlay), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    4: Seq_num(3 H1_T33 overlay), alive, sla(0x1), gid(0), cfg_order(2), local cost(0),
selected
  Src address(2):
        172.31.0.0-172.31.255.255
        10.0.3.0-10.0.3.255
  Dst address(2):
        172.31.0.0-172.31.255.255
        10.0.4.0-10.0.4.255
…
```

Since shortcut-priority is enabled, we observe that the shortcut is formed over the selected overlay path and prioritized over the parent overlay.

From the diagnostic command on Spoke 1, we observe the selected shortcut path in **bold**. (Note that the remote IP matches Spoke 2 H1_T22 in the corresponding table above.)

```
Branch1_FGT# diagnose sys sdwan advpn-session
Session head(Branch2_FGT-0-overlay:1)
(1) Service ID(1), last access(7809088), remote health check info(3)
Selected path: local(H1_T11, port1) gw: 172.31.3.1  remote IP: 172.31.3.105(172.31.81.2)
Remote information:
1: latency: 0.176267 jitter: 0.005733 pktloss: 0.000000 mos: 4.404302 sla: 0x1 cost: 0
transport_group: 1 bandwidth up: 999994 down: 999997 bidirection: 1999991
ipv4: 172.31.3.105(172.31.81.2) ipv6 2000:172:31:3::105(::)
2: latency: 0.119133 jitter: 0.004800 pktloss: 0.000000 mos: 4.404331 sla: 0x1 cost: 0
transport_group: 2 bandwidth up: 999999 down: 999997 bidirection: 1999996
ipv4: 172.31.4.101(172.31.82.2) ipv6 1410:4b02::f088:93ee:7f00:0
(c010:4b02::788a:93ee:7f00:0)
3: latency: 0.182400 jitter: 0.008800 pktloss: 0.000000 mos: 4.404295 sla: 0x1 cost: 100
transport_group: 1 bandwidth up: 999999 down: 999997 bidirection: 1999996
ipv4: 172.31.3.101(172.31.80.2) ipv6 2000:172:31:3::101(d88a:93ee:7f00:0:d88a:93ee:7f00:0)
```

From the diagnostic command on Spoke 2, we observe the selected shortcut in **bold**.

```
Branch2_FGT# diagnose sys sdwan  health-check
Health Check(HUB):
Seq(3 H1_T33): state(alive), packet-loss(0.000%) latency(0.122), jitter(0.004), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999997), bandwidth-bi(1999996) sla_map=0x1
Seq(1 H1_T11): state(alive), packet-loss(0.000%) latency(0.186), jitter(0.011), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999997), bandwidth-bi(1999996) sla_map=0x1
Seq(2 H1_T22): state(alive), packet-loss(0.000%) latency(0.180), jitter(0.005), mos(4.404),
bandwidth-up(999994), bandwidth-dw(999997), bandwidth-bi(1999991) sla_map=0x1
Seq(2 H1_T22_0): state(alive), packet-loss(0.000%) latency(0.265), jitter(0.011), mos
(4.404), bandwidth-up(999999), bandwidth-dw(999999), bandwidth-bi(1999998) sla_map=0x1
```

Scenario 2: Traffic matching SD-WAN rule 2

In this scenario, PC 1 connected to Spoke 1 initiates an ICMP ping destined for PC2 connected to Spoke 2. Therefore, this user traffic matches SD-WAN rule 2, and traffic will go through shortcut H1_T11_0 of Spoke 1 previously established in Scenario 1 above.

The local spoke generates local-out UDP packets and sends them to the hub to trigger an IKE shortcut message exchange with updated remote spoke WAN link information. The local spoke will receive this updated remote spoke WAN link information. Then the Path Manager of Spoke 1 will recalculate the best shortcut path by comparing transport group, link quality (for SLA mode), link cost, and member configuration order between Spoke 1 and Spoke 2.

For an SLA mode service, the following algorithm is followed for considering endpoints of the best shortcut path:

1. Overlays with the same transport group
2. In-SLA overlays
3. Lowest link cost overlays
4. Member configuration order as a final tiebreaker

Based on this algorithm, the Path Manager on Spoke 1 selects Spoke 1 H1_T22 because it is the first in the priority-members order for SD-WAN rule 2, it has the lowest link cost, and it is within SLA. Likewise, the Path Manager on Spoke 1 selects Spoke 2 H1_T22 since it has the lowest link cost compared to Spoke 2 H1_T11 (which has a cost of 100), it is within SLA, and has the same transport group as Spoke 1 H1_T11. Therefore, the Path Manager of Spoke 1 calculates the best shortcut path as Spoke 1 H1_T22 to Spoke 2 H1_T22.

The Path Manager will advise IKE to establish the best shortcut and add it to SD-WAN rule 2 as follows:

```
Branch1_FGT# diagnose sys sdwan service
…
Service(2): Address Mode(IPV4) flags=0x4200 use-shortcut-sla use-shortcut
 Tie break: cfg
 Shortcut priority: 1
  Gen(12), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-compare-
order
  Member sub interface(5):
    3: seq_num(2), interface(H1_T22):
       1: H1_T22_0(72)
    4: seq_num(1), interface(H1_T11):
       1: H1_T11_0(71)
  Members(5):
    1: Seq_num(2 H1_T22_0 overlay), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    2: Seq_num(1 H1_T11_0 overlay), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected, last_used=2023-12-05 14:34:07
    3: Seq_num(2 H1_T22 overlay), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    4: Seq_num(1 H1_T11 overlay), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    5: Seq_num(3 H1_T33 overlay), alive, sla(0x1), gid(0), cfg_order(2), local cost(0),
selected
  Src address(2):
        172.31.0.0-172.31.255.255
        10.0.3.0-10.0.3.255
  Dst address(2):
        172.31.0.0-172.31.255.255
        10.0.40.0-10.0.40.255
…
```

The newly selected shortcut is prioritized over the previously selected shortcut as seen in the **bolded** output above.

From the diagnostic command on Spoke 1, we observe the selected shortcut path in **bold**. (Note that the remote IP matches Spoke 2 H1_T22 in the corresponding table above.)

```
Branch1_FGT# diagnose sys sdwan advpn-session
Session head(Branch2_FGT-0-overlay:2)
(1) Service ID(1), last access(8024725), remote health check info(3)
Selected path: local(H1_T11, port1) gw: 172.31.3.1  remote IP: 172.31.3.105(172.31.81.2)
Remote information:
1: latency: 0.118267 jitter: 0.004633 pktloss: 0.000000 mos: 4.404331 sla: 0x1 cost: 0
transport_group: 2 bandwidth up: 999999 down: 999997 bidirection: 1999996
ipv4: 172.31.4.101(172.31.82.2) ipv6 180:adfb::d88a:93ee:7f00:0
(d88a:93ee:7f00:0:d88a:93ee:7f00:0)
2: latency: 0.176067 jitter: 0.006567 pktloss: 0.000000 mos: 4.404301 sla: 0x1 cost: 0
transport_group: 1 bandwidth up: 999994 down: 999997 bidirection: 1999991
ipv4: 172.31.3.105(172.31.81.2) ipv6 2000:172:31:3::105(::)
3: latency: 0.170333 jitter: 0.008133 pktloss: 0.000000 mos: 4.404302 sla: 0x1 cost: 100
transport_group: 1 bandwidth up: 999999 down: 999997 bidirection: 1999996
ipv4: 172.31.3.101(172.31.80.2) ipv6 2000:172:31:3::101(c010:4b02::788a:93ee:7f00:0)
(1) Service ID(2), last access(8024725), remote health check info(3)
Selected path: local(H1_T22, port2) gw: 172.31.3.5  remote IP: 172.31.3.105(172.31.81.2)
Remote information:
1: latency: 0.118267 jitter: 0.004633 pktloss: 0.000000 mos: 4.404331 sla: 0x1 cost: 0
transport_group: 2 bandwidth up: 999999 down: 999997 bidirection: 1999996
ipv4: 172.31.4.101(172.31.82.2) ipv6 180:adfb::d88a:93ee:7f00:0
(d88a:93ee:7f00:0:d88a:93ee:7f00:0)
2: latency: 0.176067 jitter: 0.006567 pktloss: 0.000000 mos: 4.404301 sla: 0x1 cost: 0
transport_group: 1 bandwidth up: 999994 down: 999997 bidirection: 1999991
ipv4: 172.31.3.105(172.31.81.2) ipv6 2000:172:31:3::105(::)
3: latency: 0.170333 jitter: 0.008133 pktloss: 0.000000 mos: 4.404302 sla: 0x1 cost: 100
transport_group: 1 bandwidth up: 999999 down: 999997 bidirection: 1999996
ipv4: 172.31.3.101(172.31.80.2) ipv6 2000:172:31:3::101(c010:4b02::788a:93ee:7f00:0)
…
```

From the diagnostic command on Spoke 2, we observe the selected shortcut in **bold**:

```
Branch2_FGT# diagnose sys sdwan  health-check
Health Check(HUB):
Seq(3 H1_T33): state(alive), packet-loss(0.000%) latency(0.118), jitter(0.005), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x1
Seq(1 H1_T11): state(alive), packet-loss(0.000%) latency(0.171), jitter(0.005), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x1
Seq(2 H1_T22): state(alive), packet-loss(0.000%) latency(0.175), jitter(0.006), mos(4.404),
bandwidth-up(999994), bandwidth-dw(999998), bandwidth-bi(1999992) sla_map=0x1
```

```
Seq(2 H1_T22_0): state(alive), packet-loss(0.000%) latency(0.240), jitter(0.009), mos
(4.404), bandwidth-up(1000000), bandwidth-dw(1000000), bandwidth-bi(2000000) sla_map=0x1
Seq(2 H1_T22_1): state(alive), packet-loss(0.000%) latency(0.259), jitter(0.019), mos
(4.404), bandwidth-up(1000000), bandwidth-dw(1000000), bandwidth-bi(2000000) sla_map=0x1
```

Scenario 3: Traffic matching SD-WAN rule 3

In this scenario, PC 1 connected to Spoke 1 initiates an ICMP ping destined for PC 3 connected to Spoke 2. Therefore, this user traffic matches SD-WAN rule 3, and traffic will go through shortcut H1_T11_0 of Spoke 1 previously established in Scenario 1 above.

The local spoke generates local-out UDP packets and sends them to the hub to trigger an IKE shortcut message exchange with updated remote spoke WAN link information. The local spoke will receive this updated remote spoke WAN link information. Then the Path Manager of Spoke 1 will recalculate the best shortcut path by comparing transport group, best quality (based on link cost factor), and member configuration order between Spoke 1 and Spoke 2.

For a best quality mode service, the following algorithm is followed for considering endpoints of the best shortcut path:

1. Overlays with the same transport group
2. Best quality overlays (link cost factor of packet loss, in this scenario)
3. Member configuration order as a final tiebreaker

Based on this algorithm, the Path Manager on Spoke 1 selects Spoke 1 H1_T33 because it is the first in the priority-members order for SD-WAN rule 3, and it has the best quality link. Likewise, the Path Manager on Spoke 1 selects Spoke 2 H1_T33 since it has the same transport group as Spoke 1 H1_T33. Therefore, the Path Manager of Spoke 1 calculates the best shortcut path as Spoke 1 H1_T33 to Spoke 2 H1_T33.

The Path Manager will advise IKE to establish the best shortcut and add it to SD-WAN rule 3 as follows:

```
Branch1_FGT# diagnose sys sdwan service
…
Service(3): Address Mode(IPV4) flags=0x4200 use-shortcut-sla use-shortcut
 Tie break: cfg
 Shortcut priority: 3
  Gen(13), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(priority), link-
cost-factor(packet-loss), link-cost-threshold(10), heath-check(HUB)
  Member sub interface(6):
    4: seq_num(3), interface(H1_T33):
       1: H1_T33_0(73)
    5: seq_num(1), interface(H1_T11):
       1: H1_T11_0(71)
    6: seq_num(2), interface(H1_T22):
       1: H1_T22_0(72)
  Members(6):
    1: Seq_num(3 H1_T33_0 overlay), alive, packet loss: 0.000%, selected
    2: Seq_num(1 H1_T11_0 overlay), alive, packet loss: 0.000%, selected, last_used=2023-12-
05 14:38:02
    3: Seq_num(2 H1_T22_0 overlay), alive, packet loss: 0.000%, selected
    4: Seq_num(3 H1_T33 overlay), alive, packet loss: 0.000%, selected
    5: Seq_num(1 H1_T11 overlay), alive, packet loss: 0.000%, selected
    6: Seq_num(2 H1_T22 overlay), alive, packet loss: 0.000%, selected
  Src address(2):
        172.31.0.0-172.31.255.255
        10.0.3.0-10.0.3.255
  Dst address(2):
        172.31.0.0-172.31.255.255
        10.0.41.0-10.0.41.255
```

From the diagnostic command on Spoke 1, we observe the selected shortcut path in **bold**. (Note that the remote IP matches Spoke 2 H1_T33 in the corresponding table above.)

```
Branch1_FGT# diagnose sys sdwan advpn-session
Session head(Branch2_FGT-0-overlay:3)
(1) Service ID(3), last access(8047297), remote health check info(3)
Selected path: local(H1_T33, port3) gw: 172.31.4.1  remote IP: 172.31.4.101(172.31.82.2)
Remote information:
1: latency: 0.116600 jitter: 0.004600 pktloss: 0.000000 mos: 4.404332 sla: 0x1 cost: 0
transport_group: 2 bandwidth up: 999999 down: 999998 bidirection: 1999997
ipv4: 172.31.4.101(172.31.82.2) ipv6 180:adfb::d88a:93ee:7f00:0
(d88a:93ee:7f00:0:d88a:93ee:7f00:0)
2: latency: 0.174767 jitter: 0.005533 pktloss: 0.000000 mos: 4.404303 sla: 0x1 cost: 0
transport_group: 1 bandwidth up: 999994 down: 999998 bidirection: 1999992
ipv4: 172.31.3.105(172.31.81.2) ipv6 2000:172:31:3::105(c010:4b02::788a:93ee:7f00:0)
3: latency: 0.172900 jitter: 0.005167 pktloss: 0.000000 mos: 4.404304 sla: 0x1 cost: 100
transport_group: 1 bandwidth up: 999999 down: 999998 bidirection: 1999997
ipv4: 172.31.3.101(172.31.80.2) ipv6 2000:172:31:3::101(::)
```

From the diagnostic command on Spoke 2, we observe the selected shortcut in **bold**:

```
Branch2_FGT# diagnose sys sdwan  health-check
Health Check(HUB):
Seq(3 H1_T33): state(alive), packet-loss(0.000%) latency(0.116), jitter(0.005), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x1
Seq(3 H1_T33_0): state(alive), packet-loss(0.000%) latency(0.113), jitter(0.005), mos
(4.404), bandwidth-up(1000000), bandwidth-dw(1000000), bandwidth-bi(2000000) sla_map=0x1
Seq(1 H1_T11): state(alive), packet-loss(0.000%) latency(0.171), jitter(0.004), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x1
Seq(2 H1_T22): state(alive), packet-loss(0.000%) latency(0.174), jitter(0.008), mos(4.404),
bandwidth-up(999994), bandwidth-dw(999998), bandwidth-bi(1999992) sla_map=0x1
Seq(2 H1_T22_0): state(alive), packet-loss(0.000%) latency(0.239), jitter(0.007), mos
(4.404), bandwidth-up(999999), bandwidth-dw(999999), bandwidth-bi(1999998) sla_map=0x1
Seq(2 H1_T22_1): state(alive), packet-loss(0.000%) latency(0.260), jitter(0.014), mos
(4.404), bandwidth-up(1000000), bandwidth-dw(1000000), bandwidth-bi(2000000) sla_map=0x1
```

Scenario 4: Spoke 2 H1_T22 overlay link out-of-SLA

In this scenario, we place remote Spoke 2 H1_T22 out-of-SLA and observe that this link quality change is sensed by the local spoke through regular WAN link information updates on shortcuts. Then the local Spoke 1 will generate local-out UDP packets and send them to the hub to trigger an IKE shortcut message exchange. Once Spoke 1 receives a shortcut reply, it will start to calculate new best shortcut paths for SD-WAN rules 1 and 2 because these are the only rules that have new best shortcut paths when Spoke 2 H1_T22 is out-of-SLA.

For an SLA mode service, the following algorithm is followed for considering endpoints of the best shortcut path:

1. Overlays with the same transport group
2. In-SLA overlays
3. Lowest link cost overlays
4. Member configuration order as a final tiebreaker

Based on this algorithm, the Path Manager on Spoke 1 still selects these Spoke 1 interfaces:

- SD-WAN Rule 1: H1_T11
- SD-WAN Rule 2: H1_T22

These are the first in the priority-members order for SD-WAN rules 1 and 2, respectively.

Based on the updated WAN link information, the Path Manager on Spoke 1 selects these Spoke 2 interfaces because they are the only remaining in-SLA VPN overlays over Internet links (transport group 1):

- SD-WAN Rule 1: H1_T11
- SD-WAN Rule 2: H1_T11

Therefore, the Path Manager of Spoke 1 calculates the best shortcut paths as follows:

- SD-WAN Rule 1: Spoke 1 H1_T11 to Spoke 2 H1_T11
- SD-WAN Rule 2: Spoke 1 H1_T22 to Spoke 2 H1_T11

The Path Manager will advise IKE to establish the best shortcuts and add them to SD-WAN rules 1 and 2 as follows:

- For SD-WAN Rule 1, H1_T11_1 is the new best shortcut.
- For SD-WAN Rule 2, H1_T22_1 is the new best shortcut.

```
# diagnose sys sdwan service
Service(1): Address Mode(IPV4) flags=0x4200 use-shortcut-sla use-shortcut
 Tie break: cfg
 Shortcut priority: 1
  Gen(17), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-compare-
order
  Member sub interface(8):
    6: seq_num(1), interface(H1_T11):
       1: H1_T11_0(74)
       2: H1_T11_1(75)
    7: seq_num(2), interface(H1_T22):
       1: H1_T22_0(72)
       2: H1_T22_1(76)
    8: seq_num(3), interface(H1_T33):
       1: H1_T33_0(73)
  Members(8):
    1: Seq_num(1 H1_T11_0 overlay), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    2: Seq_num(1 H1_T11_1 overlay), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    3: Seq_num(2 H1_T22_0 overlay), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    4: Seq_num(2 H1_T22_1 overlay), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    5: Seq_num(3 H1_T33_0 overlay), alive, sla(0x1), gid(0), cfg_order(2), local cost(0),
selected
    6: Seq_num(1 H1_T11 overlay), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    7: Seq_num(2 H1_T22 overlay), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    8: Seq_num(3 H1_T33 overlay), alive, sla(0x1), gid(0), cfg_order(2), local cost(0),
selected
  Src address(2):
        172.31.0.0-172.31.255.255
        10.0.3.0-10.0.3.255
  Dst address(2):
        172.31.0.0-172.31.255.255
        10.0.4.0-10.0.4.255

Service(2): Address Mode(IPV4) flags=0x4200 use-shortcut-sla use-shortcut
 Tie break: cfg
 Shortcut priority: 1
  Gen(17), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-compare-
order
  Member sub interface(8):
    6: seq_num(2), interface(H1_T22):
       1: H1_T22_0(72)
       2: H1_T22_1(76)
    7: seq_num(1), interface(H1_T11):
       1: H1_T11_0(74)
       2: H1_T11_1(75)
    8: seq_num(3), interface(H1_T33):
       1: H1_T33_0(73)
  Members(8):
    1: Seq_num(2 H1_T22_0 overlay), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    2: Seq_num(2 H1_T22_1 overlay), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    3: Seq_num(1 H1_T11_1 overlay), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    4: Seq_num(1 H1_T11_0 overlay), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    5: Seq_num(3 H1_T33_0 overlay), alive, sla(0x1), gid(0), cfg_order(2), local cost(0),
selected
    6: Seq_num(2 H1_T22 overlay), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    7: Seq_num(1 H1_T11 overlay), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    8: Seq_num(3 H1_T33 overlay), alive, sla(0x1), gid(0), cfg_order(2), local cost(0),
selected
  Src address(2):
        172.31.0.0-172.31.255.255
        10.0.3.0-10.0.3.255
  Dst address(2):
        172.31.0.0-172.31.255.255
        10.0.40.0-10.0.40.255
…
```

From the diagnostic command on Spoke 1, we observe the newly selected shortcut paths in **bold**. (Note that the remote IP 172.31.80.2 matches Spoke 2 H1_T11, which is the VPN overlay over the Internet link with cost 100 in the corresponding table above.)

```
# diagnose sys sdwan advpn-session
Session head(Branch2_FGT-0-overlay:3)
(1) Service ID(1), last access(8293060), remote health check info(3)
Selected path: local(H1_T11, port1) gw: 172.31.3.1  remote IP: 172.31.3.101(172.31.80.2)
Remote information:
1: latency: 0.119500 jitter: 0.006067 pktloss: 0.000000 mos: 4.404329 sla: 0x1 cost: 0
transport_group: 2 bandwidth up: 999999 down: 999997 bidirection: 1999996
ipv4: 172.31.4.101(172.31.82.2) ipv6 180:adfb::d88a:93ee:7f00:0
(d88a:93ee:7f00:0:d88a:93ee:7f00:0)
2: latency: 250.170761 jitter: 0.011500 pktloss: 0.000000 mos: 3.992655 sla: 0x0 cost: 0
transport_group: 1 bandwidth up: 999994 down: 999997 bidirection: 1999991
ipv4: 172.31.3.105(172.31.81.2) ipv6 2000:172:31:3::105(c010:4b02::788a:93ee:7f00:0)
```

```
3: latency: 0.182200 jitter: 0.012000 pktloss: 0.000000 mos: 4.404292 sla: 0x1 cost: 100
transport_group: 1 bandwidth up: 999999 down: 999997 bidirection: 1999996
ipv4: 172.31.3.101(172.31.80.2) ipv6 2000:172:31:3::101(::)
(1) Service ID(2), last access(8293060), remote health check info(3)
Selected path: local(H1_T22, port2) gw: 172.31.3.5  remote IP: 172.31.3.101(172.31.80.2)
Remote information:
1: latency: 0.119500 jitter: 0.006067 pktloss: 0.000000 mos: 4.404329 sla: 0x1 cost: 0
transport_group: 2 bandwidth up: 999999 down: 999997 bidirection: 1999996
ipv4: 172.31.4.101(172.31.82.2) ipv6 180:adfb::d88a:93ee:7f00:0
(d88a:93ee:7f00:0:d88a:93ee:7f00:0)
2: latency: 250.170761 jitter: 0.011500 pktloss: 0.000000 mos: 3.992655 sla: 0x0 cost: 0
transport_group: 1 bandwidth up: 999994 down: 999997 bidirection: 1999991
ipv4: 172.31.3.105(172.31.81.2) ipv6 2000:172:31:3::105(c010:4b02::788a:93ee:7f00:0)
3: latency: 0.182200 jitter: 0.012000 pktloss: 0.000000 mos: 4.404292 sla: 0x1 cost: 100
transport_group: 1 bandwidth up: 999999 down: 999997 bidirection: 1999996
ipv4: 172.31.3.101(172.31.80.2) ipv6 2000:172:31:3::101(::)
```

From the diagnostic command on Spoke 2, we observe the selected shortcuts in **bold**:

```
Branch2_FGT# diagnose sys sdwan  health-check
Health Check(HUB):
Seq(3 H1_T33): state(alive), packet-loss(0.000%) latency(0.120), jitter(0.007), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999997), bandwidth-bi(1999996) sla_map=0x1
Seq(3 H1_T33_0): state(alive), packet-loss(0.000%) latency(0.128), jitter(0.003), mos
(4.404), bandwidth-up(1000000), bandwidth-dw(1000000), bandwidth-bi(2000000) sla_map=0x1
Seq(1 H1_T11): state(alive), packet-loss(0.000%) latency(0.180), jitter(0.008), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999997), bandwidth-bi(1999996) sla_map=0x1
Seq(1 H1_T11_0): state(alive), packet-loss(0.000%) latency(0.259), jitter(0.023), mos
(4.404), bandwidth-up(1000000), bandwidth-dw(1000000), bandwidth-bi(2000000) sla_map=0x1
Seq(1 H1_T11_1): state(alive), packet-loss(0.000%) latency(0.257), jitter(0.014), mos
(4.404), bandwidth-up(1000000), bandwidth-dw(1000000), bandwidth-bi(2000000) sla_map=0x1
Seq(2 H1_T22): state(alive), packet-loss(0.000%) latency(250.169), jitter(0.009), mos
(3.993), bandwidth-up(999994), bandwidth-dw(999997), bandwidth-bi(1999991) sla_map=0x0
Seq(2 H1_T22_1): state(alive), packet-loss(0.000%) latency(0.245), jitter(0.013), mos
(4.404), bandwidth-up(1000000), bandwidth-dw(1000000), bandwidth-bi(2000000) sla_map=0x1
Seq(2 H1_T22_0): state(alive), packet-loss(0.000%) latency(0.223), jitter(0.005), mos
(4.404), bandwidth-up(1000000), bandwidth-dw(1000000), bandwidth-bi(2000000) sla_map=0x1
```

Monitoring

7.4.0

7.4.1

7.4.2

Logging FortiMonitor-detected performance metrics

This information is also available in the FortiOS 7.4 Administration Guide:

- SD-WAN application monitor using FortiMonitor

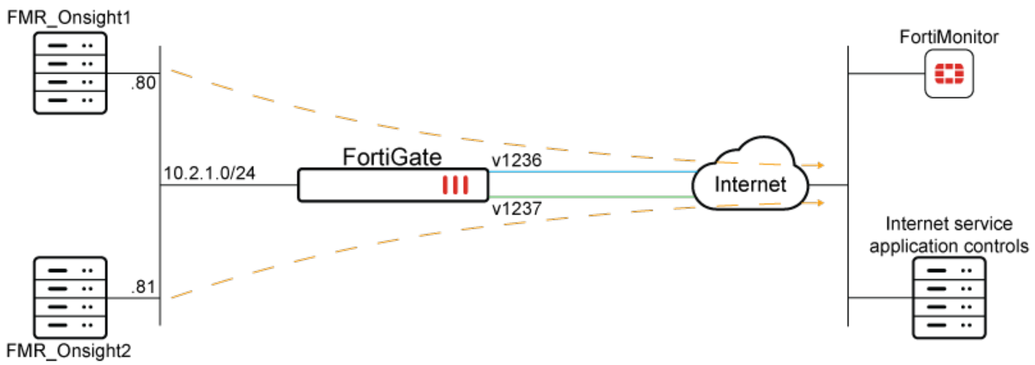
FortiGate can log statistics when using FortiMonitor to detect advanced SD-WAN application performance metrics. These logs may also be sent to FortiAnalyzer and FortiManager for review and reporting.

You can control the logging frequency using the `app-perf-log-period` command:

```
config system sdwan
    set app-perf-log-period <time in seconds>
end
```

Example

This example is based on the following topology:



To configure logging of FortiMonitor-detected performance metrics:

1. Configure the address objects for each FortiMonitor client:

```
config firewall address
    edit "FMR_OnSight1"
        set subnet 10.2.1.80 255.255.255.255
    next
    edit "FMR_OnSight2"
        set subnet 10.2.1.81 255.255.255.255
    next
end
```

2. Set the logging frequency:

```
config system sdwan
    set status enable
    set app-perf-log-period 60
end
```

3. Configure the SD-WAN zone and members:

```
config system sdwan
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "v1236"
            set gateway 10.12.36.2
        next
        edit 2
            set interface "v1237"
            set gateway 10.12.37.20
        next
    end
end
```

4. Configure the SD-WAN rules:

```
config system sdwan
    config service
        edit 1
            set dst "all"
            set src "FMR_OnSight1"
            set priority-members 2
            set agent-exclusive enable
        next
        edit 2
            set dst "all"
            set src "FMR_OnSight2"
            set priority-members 1
            set agent-exclusive enable
        next
    end
end
```

5. Configure the SD-WAN health check:

```
config system sdwan
    config health-check
        edit "FMR"
            set detect-mode agent-based
            set probe-timeout 60000
            set recoverytime 1
            set members 1 2
            config sla
                edit 1
                next
            end
        next
    end
end
```

To verify SD-WAN member performance and review logs:

1. Verify the health check diagnostics:

```
# diagnose sys sdwan health-check
    Health Check(FMR):
    Seq(1 v1236): state(alive), packet-loss(0.000%) latency(200.099), jitter(0.201), mos
(4.171), bandwidth-up(999989), bandwidth-dw(999983), bandwidth-bi(1999972) sla_map=0x0
    Seq(2 v1237): state(alive), packet-loss(0.000%) latency(200.103), jitter(0.391), mos
(4.169), bandwidth-up(999994), bandwidth-dw(999981), bandwidth-bi(1999975) sla_map=0x0
```

2. Review the SD-WAN logs:

```
# execute log filter category event
# execute log filter field subtype sdwan
# execute log display

1: date=2023-01-27 time=16:32:15 eventtime=1674865935918381398 tz="-0800"
logid="0113022937" type="event" subtype="sdwan" level="information" vd="root"
logdesc="Virtuan WAN Link application performance metrics via FortiMonitor"
eventtype="Application Performance Metrics" app="fortinet.com" appid=0 interface="v1237"
latency="200.2" jitter="0.6" packetloss="0.0" serverresponsetime="827.7"
networktransfertime="107.7" apperror="0.0" timestamp="01-28 00:31:59" msg="Application
Performance Metrics via FortiMonitor"

2: date=2023-01-27 time=16:32:15 eventtime=1674865935918367770 tz="-0800"
logid="0113022937" type="event" subtype="sdwan" level="information" vd="root"
logdesc="Virtuan WAN Link application performance metrics via FortiMonitor"
```

```
eventtype="Application Performance Metrics" app="fortinet.com" appid=0 interface="v1236"
latency="200.0" jitter="0.3" packetloss="0.0" serverresponsetime="870.6"
networktransfertime="130.4" apperror="0.0" timestamp="01-28 00:31:59" msg="Application
Performance Metrics via FortiMonitor"

3: date=2023-01-27 time=16:31:15 eventtime=1674865875917685437 tz="-0800"
logid="0113022937" type="event" subtype="sdwan" level="information" vd="root"
logdesc="Virtuan WAN Link application performance metrics via FortiMonitor"
eventtype="Application Performance Metrics" app="fortinet.com" appid=0 interface="v1237"
latency="200.5" jitter="0.7" packetloss="0.0" serverresponsetime="1008.9"
networktransfertime="129.8" apperror="0.0" timestamp="01-28 00:31:02" msg="Application
Performance Metrics via FortiMonitor"

4: date=2023-01-27 time=16:31:15 eventtime=1674865875917672824 tz="-0800"
logid="0113022937" type="event" subtype="sdwan" level="information" vd="root"
logdesc="Virtuan WAN Link application performance metrics via FortiMonitor"
eventtype="Application Performance Metrics" app="fortinet.com" appid=0 interface="v1236"
latency="200.3" jitter="0.8" packetloss="0.0" serverresponsetime="825.4"
networktransfertime="106.4" apperror="0.0" timestamp="01-28 00:31:02" msg="Application
Performance Metrics via FortiMonitor"

5: date=2023-01-27 time=16:30:15 eventtime=1674865815912801725 tz="-0800"
logid="0113022937" type="event" subtype="sdwan" level="information" vd="root"
logdesc="Virtuan WAN Link application performance metrics via FortiMonitor"
eventtype="Application Performance Metrics" app="fortinet.com" appid=0 interface="v1237"
latency="200.1" jitter="0.4" packetloss="0.0" serverresponsetime="845.4"
networktransfertime="116.0" apperror="0.0" timestamp="01-28 00:30:01" msg="Application
Performance Metrics via FortiMonitor"

6: date=2023-01-27 time=16:30:15 eventtime=1674865815912786458 tz="-0800"
logid="0113022937" type="event" subtype="sdwan" level="information" vd="root"
logdesc="Virtuan WAN Link application performance metrics via FortiMonitor"
eventtype="Application Performance Metrics" app="fortinet.com" appid=0 interface="v1236"
latency="200.0" jitter="0.3" packetloss="0.0" serverresponsetime="1032.0"
networktransfertime="138.9" apperror="0.0" timestamp="01-28 00:30:01" msg="Application
Performance Metrics via FortiMonitor"
```

SD-WAN Monitoring Map integrates with Cloud Assisted Monitoring Service to allow FGT interface speed tests from inside FMG - FMG

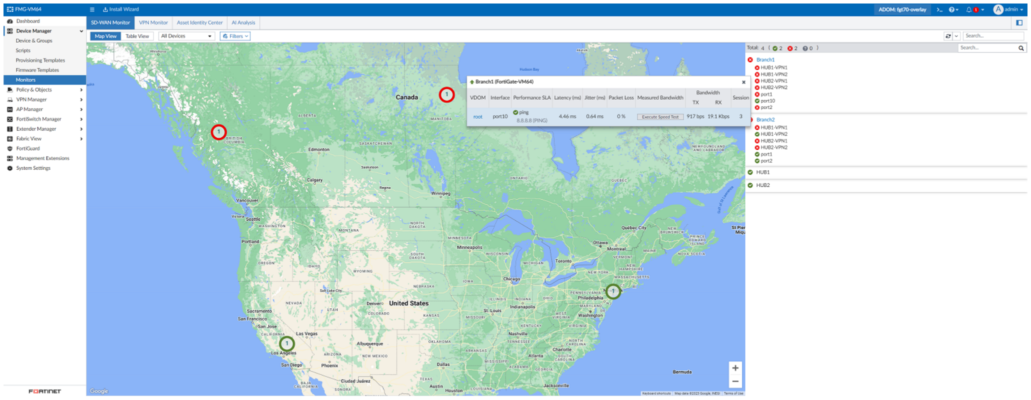
This information is also available in the FortiManager 7.4 Administration Guide:

- SD-WAN Cloud assisted monitoring speed test.

SD-WAN Monitoring Map integrates with Cloud Assisted Monitoring Service to allow FortiGate interface speed tests from inside FortiManager.

To execute an SD-WAN speed test:

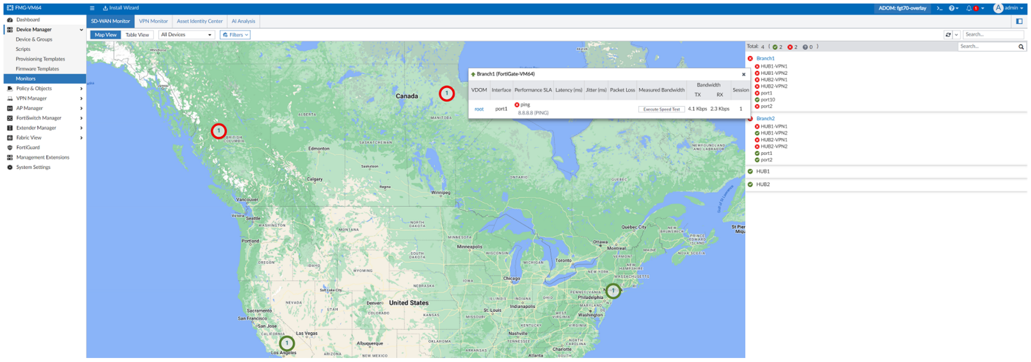
1. Execution of speed tests can be performed from the SD-WAN Monitor page: Map View, Table View, Device Drilldown and the Device Dashboard.
2. For devices with a valid license and an interface set with the WAN role, the *Execute Speed Test* option is displayed for the interface.
   - If there is a valid route to the cloud server, you will get measured bandwidth when executing the speed test.



Branch1 (FortiGate-VM64)

VDOM	Interface	Performance SLA	Latency (ms)	Jitter (ms)	Packet Loss	Measured Bandwidth	Bandwidth TX	Bandwidth RX	Session
root	port10	ping 8.8.8.8 (PING)	4.46 ms	0.64 ms	0 %	Execute Speed Test	917 bps	19.1 Kbps	3

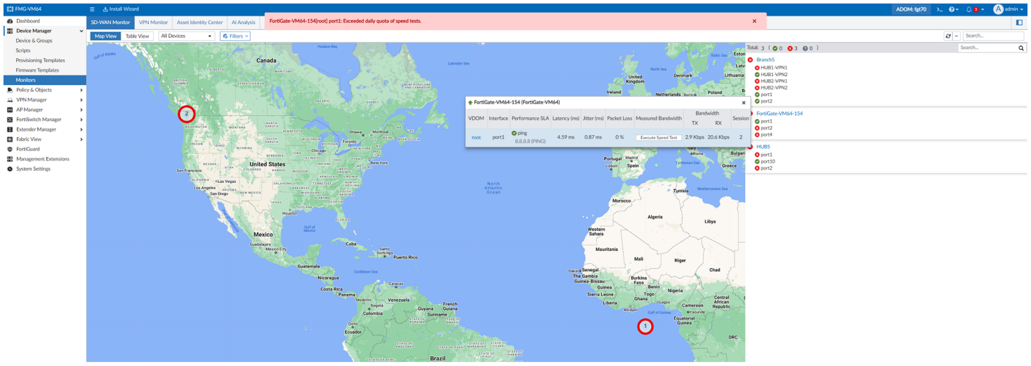
   - If there is not a valid route to the cloud server, you will see an error message when executing the speed test.



Branch1 (FortiGate-VM64)

VDOM	Interface	Performance SLA	Latency (ms)	Jitter (ms)	Packet Loss	Measured Bandwidth	Bandwidth TX	Bandwidth RX	Session
root	port1	ping 8.8.8.8 (PING)				Execute Speed Test	4.1 Kbps	2.3 Kbps	1

   - You can perform the speed test up to 10 times per day. Attempts to perform additional speed tests will present an error message.

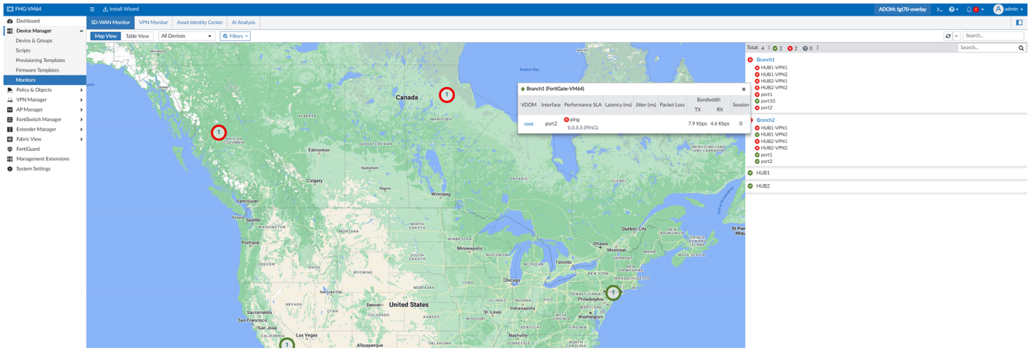


FortiGate-VM64-154[root] port1: Exceeded daily quota of speed tests.

FortiGate-VM64-154 (FortiGate-VM64)

VDOM	Interface	Performance SLA	Latency (ms)	Jitter (ms)	Packet Loss	Measured Bandwidth	Bandwidth TX	Bandwidth RX	Session
root	port1	ping 8.8.8.8 (PING)	4.59 ms	0.87 ms	0 %	Execute Speed Test	2.9 Kbps	20.6 Kbps	2

3. For devices without a valid license, or for devices with a valid license but without an interface set to the WAN role, the *Execute Speed Test* option is not displayed.



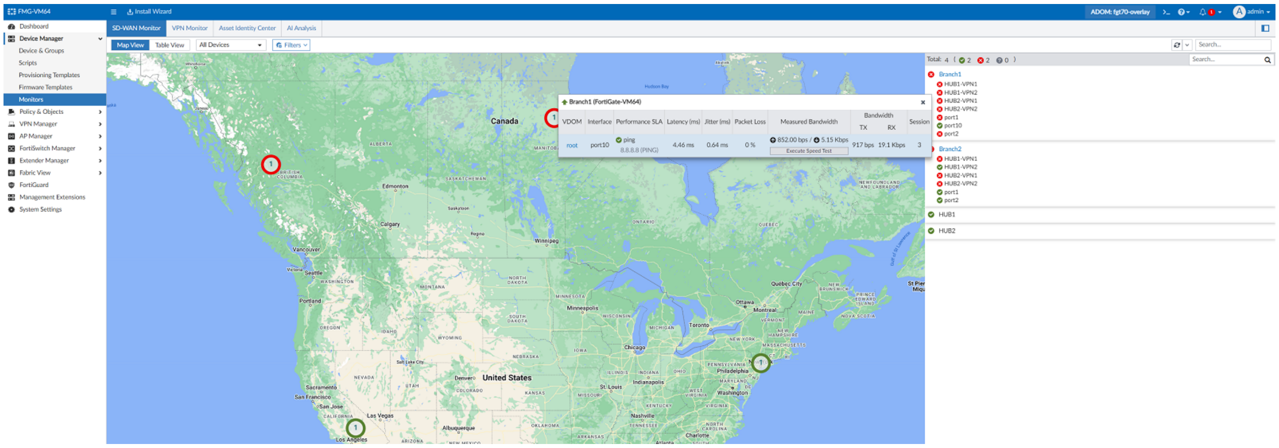
Branch1 (FortiGate-VM64)

VDOM	Interface	Performance SLA	Latency (ms)	Jitter (ms)	Packet Loss	Bandwidth		Session
						TX	RX	
root	port2	ping 8.8.8.8 (PING)				7.9 Kbps	4.6 Kbps	0

To view the results in SD-WAN Monitor pages:

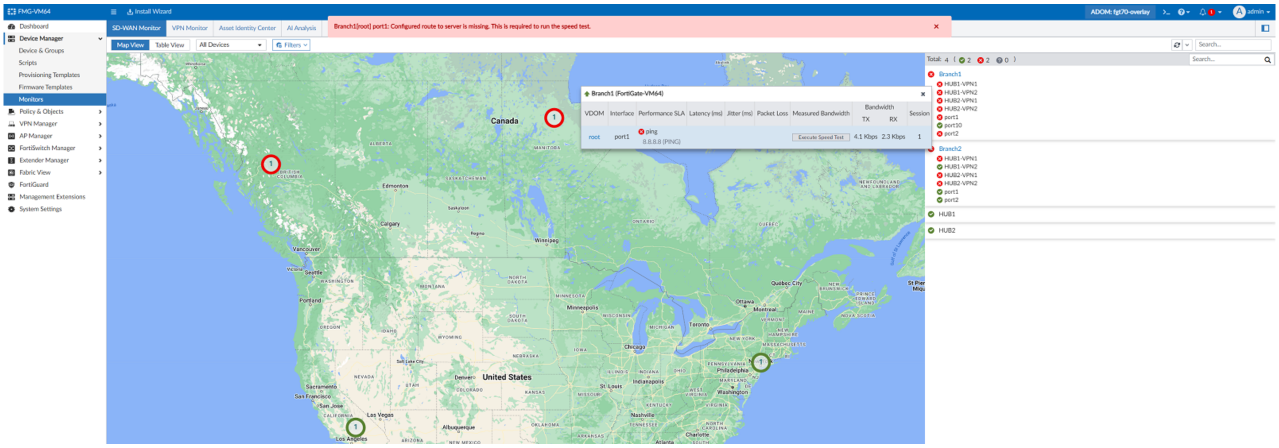
The latest results of the speed test are displayed on the SD-WAN Monitor pages, including:

- Map View:



Branch1 (FortiGate-VM64)

VDOM	Interface	Performance SLA	Latency (ms)	Jitter (ms)	Packet Loss	Measured Bandwidth	Bandwidth		Session
							TX	RX	
root	port10	ping 8.8.8.8 (PING)	4.46 ms	0.64 ms	0 %	852.00 bps / 5.15 Kbps Execute Speed Test	917 bps	19.1 Kbps	3

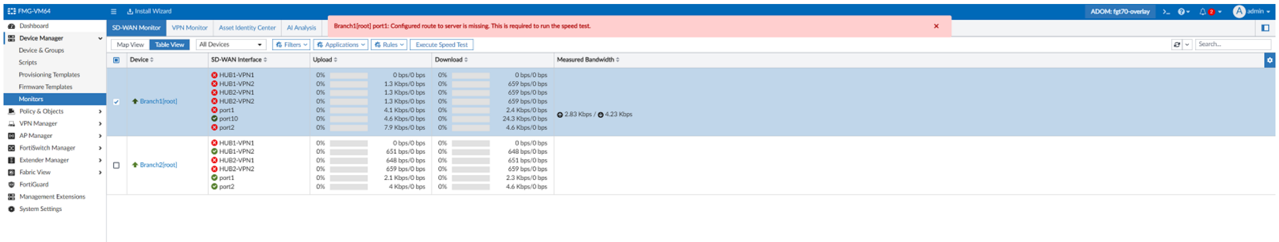


Branch1[root] port1: Configured route to server is missing. This is required to run the speed test.

Branch1 (FortiGate-VM64)

VDOM	Interface	Performance SLA	Latency (ms)	Jitter (ms)	Packet Loss	Measured Bandwidth	Bandwidth		Session
							TX	RX	
root	port1	ping 8.8.8.8 (PING)				Execute Speed Test	4.1 Kbps	2.3 Kbps	1

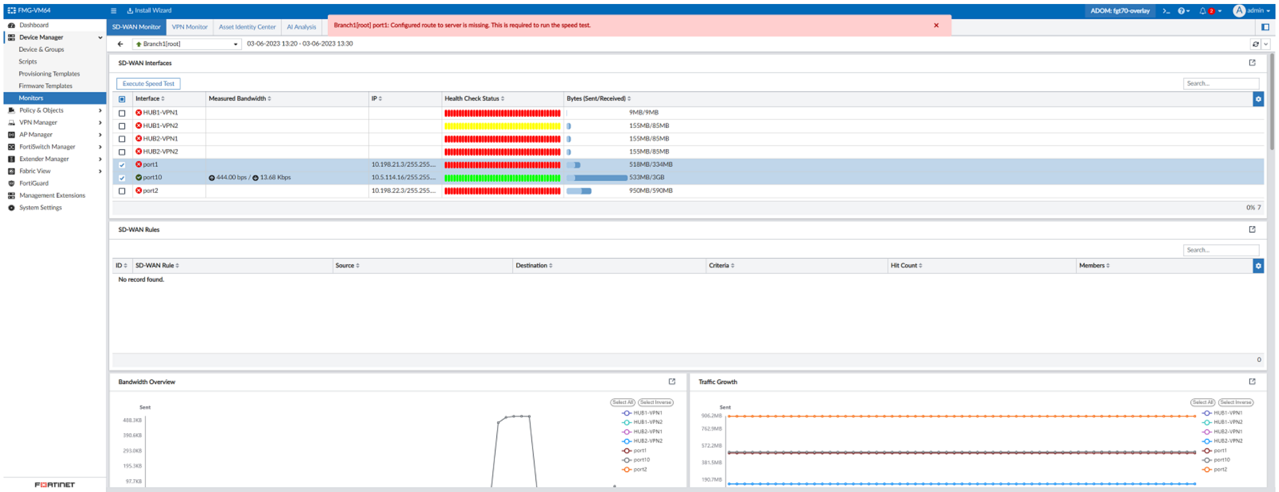
- Table View:



Branch1[root] port1: Configured route to server is missing. This is required to run the speed test.

	Device	SD-WAN Interface	Upload		Download		Measured Bandwidth
☑	Branch1[root]	HUB1-VPN1	0%	0 bps/0 bps	0%	0 bps/0 bps	2.83 Kbps / 4.23 Kbps
		HUB1-VPN2	0%	1.3 Kbps/0 bps	0%	659 bps/0 bps	
		HUB2-VPN1	0%	1.3 Kbps/0 bps	0%	659 bps/0 bps	
		HUB2-VPN2	0%	1.3 Kbps/0 bps	0%	659 bps/0 bps	
		port1	0%	4.1 Kbps/0 bps	0%	2.4 Kbps/0 bps	
		port10	0%	4.6 Kbps/0 bps	0%	24.3 Kbps/0 bps	
		port2	0%	7.9 Kbps/0 bps	0%	4.6 Kbps/0 bps	
☐	Branch2[root]	HUB1-VPN1	0%	0 bps/0 bps	0%	0 bps/0 bps	
		HUB1-VPN2	0%	651 bps/0 bps	0%	648 bps/0 bps	
		HUB2-VPN1	0%	648 bps/0 bps	0%	651 bps/0 bps	
		HUB2-VPN2	0%	659 bps/0 bps	0%	659 bps/0 bps	
		port1	0%	2.1 Kbps/0 bps	0%	2.3 Kbps/0 bps	
		port2	0%	4 Kbps/0 bps	0%	4.6 Kbps/0 bps	

- Device Drilldown:



Branch1[root] port1: Configured route to server is missing. This is required to run the speed test.

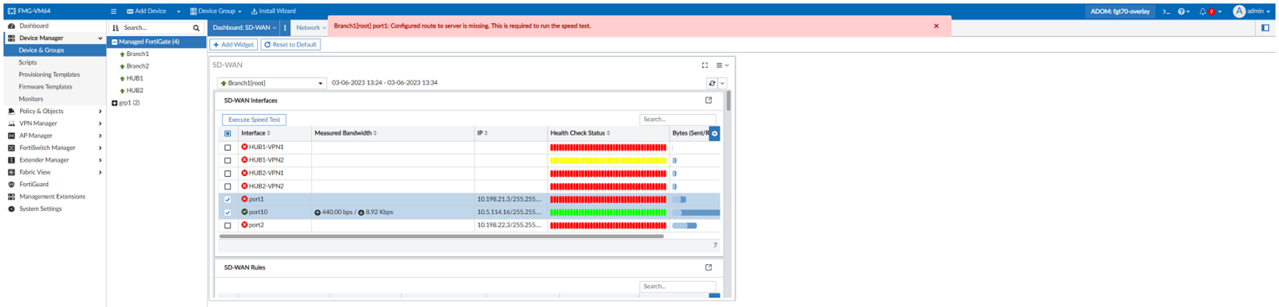
03-06-2023 13:20 - 03-06-2023 13:30

	Interface	Measured Bandwidth	IP	Health Check Status	Bytes (Sent/Received)
☐	HUB1-VPN1				9MB/9MB
☐	HUB1-VPN2				155MB/85MB
☐	HUB2-VPN1				155MB/85MB
☐	HUB2-VPN2				155MB/85MB
☑	port1		10.198.21.3/255.255....		518MB/334MB
☑	port10	444.00 bps / 13.68 Kbps	10.5.114.16/255.255....		533MB/3GB
☐	port2		10.198.22.3/255.255....		950MB/590MB

ID	SD-WAN Rule	Source	Destination	Criteria	Hit Count	Members

No record found.

- Device Manager > Device Dashboard > SD-WAN Monitor:



Branch1[root] port1: Configured route to server is missing. This is required to run the speed test.

03-06-2023 13:24 - 03-06-2023 13:34

	Interface	Measured Bandwidth	IP	Health Check Status	Bytes (Sent/R
☐	HUB1-VPN1				
☐	HUB1-VPN2				
☐	HUB2-VPN1				
☐	HUB2-VPN2				
☑	port1		10.198.21.3/255.255....		
☑	port10	440.00 bps / 8.92 Kbps	10.5.114.16/255.255....		
☐	port2		10.198.22.3/255.255....		

SD-WAN monitoring map enhancements - FMG

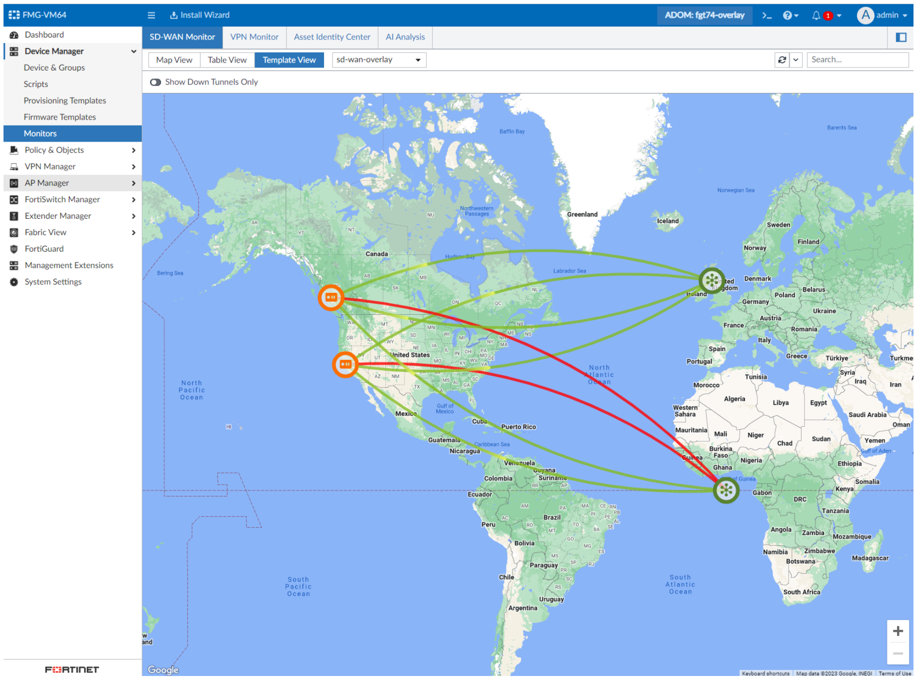
This information is also available in the FortiManager 7.4 Administration Guide:

- Template View

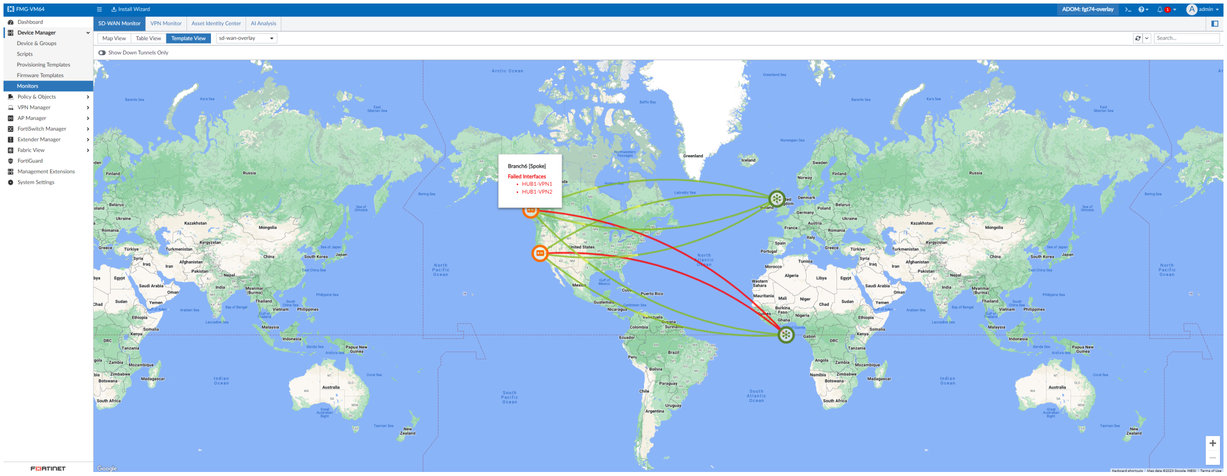
SD-WAN monitoring map differentiates HUB and branch device types, displays the overlay connectivity between devices and WAN underlay ports SLA performances.

To monitor SD-WAN with the Template View:

1. Go to the *Device Manager* > *Monitors* > *SD-WAN Monitor* pane, and click *Template View*.
   - SD-WAN devices provisioned using the currently selected SD-WAN template are displayed on the map.
   - Only devices provisioned using the selected SD-WAN template are displayed. You can change the selected SD-WAN template by clicking the dropdown in the toolbar and selecting a new template.
   - Devices on the map are identified with icons as either a HUB (star icon) or spoke device (device icon).

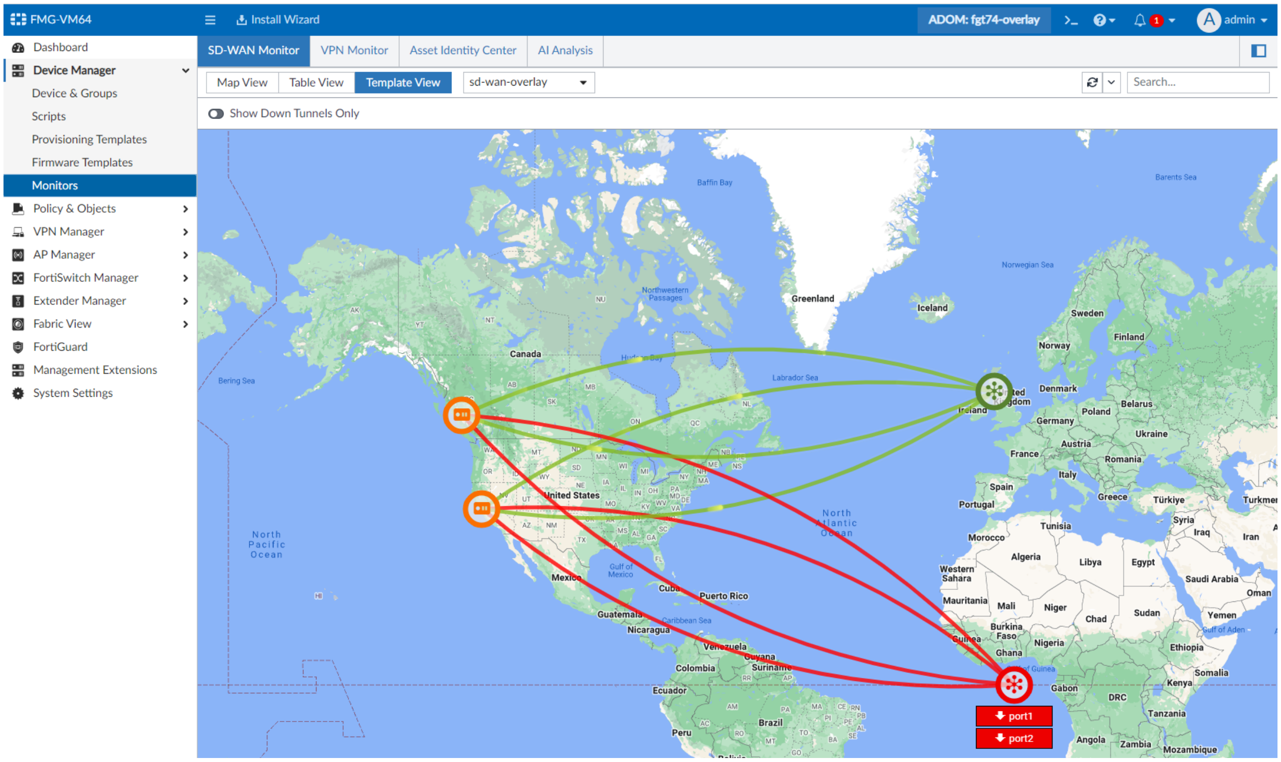


2. Hovering your mouse over a device on the map displays the following information:
   - Device name and whether it is a HUB or spoke.
   - Interfaces that have a failed health check.
   - Down underlays.


Branch6 [Spoke]

Failed Interfaces
- HUB1-VPN1
- HUB1-VPN2

3. The map shows lines connecting the HUB and spoke devices.

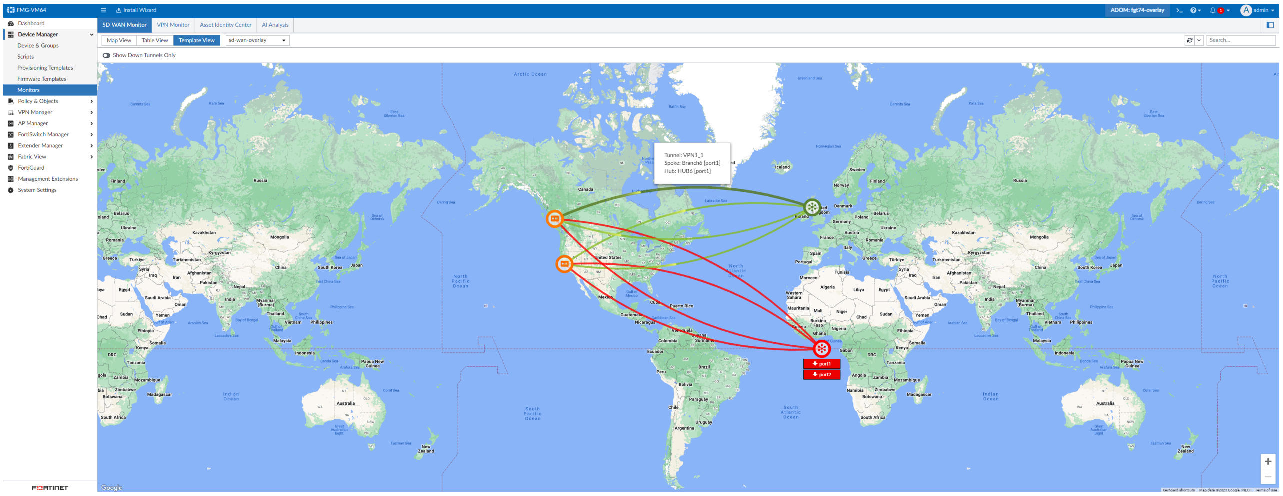


The line color depends on if the tunnel is up (green) or down (red). Device color is based off of the following logic:

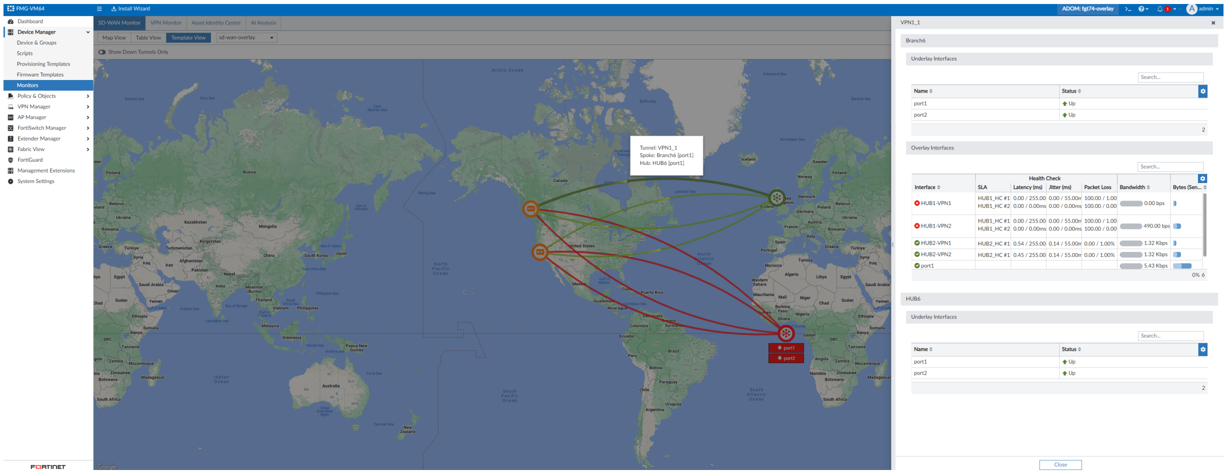
a. If the SD-WAN health checks are defined on the device (usually a spoke):
   - Green: All health checks pass.
   - Orange: Some health checks pass.
   - Red: All health checks fail.

b. When no SD-WAN health checks are defined on the device (usually a HUB):
   - Green: All underlays are up.
   - Orange: Some underlays are up.
   - Red: All underlays are down.

4. Hovering over a line displays a tooltip showing both device names.


Tunnel: VPN1_1
Spoke: Branch6 [port1]
Hub: HUB6 [port1]

5. Clicking on a line opens a pane with the following information:
   - Underlay Status table of HUB and spoke devices.
   - Health check table for the spoke devices.



VPN1_1

Branch6

Underlay Interfaces

Name	Status
port1	Up
port2	Up

2

Overlay Interfaces

Interface	Health Check				Bandwidth	Bytes (Sen...
	SLA	Latency (ms)	Jitter (ms)	Packet Loss		
HUB1-VPN1	HUB1_HC #1 HUB1_HC #2	0.00 / 255.00 0.00 / 0.00ms	0.00 / 55.00m 0.00 / 0.00ms	100.00 / 1.00 100.00 / 0.00	0.00 bps	
HUB1-VPN2	HUB1_HC #1 HUB1_HC #2	0.00 / 255.00 0.00 / 0.00ms	0.00 / 55.00m 0.00 / 0.00ms	100.00 / 1.00 100.00 / 0.00	490.00 bps	
HUB2-VPN1	HUB2_HC #1	0.54 / 255.00	0.14 / 55.00m	0.00 / 1.00%	1.32 Kbps	
HUB2-VPN2	HUB2_HC #1	0.45 / 255.00	0.14 / 55.00m	0.00 / 1.00%	1.32 Kbps	
port1					5.43 Kbps	

0% 6

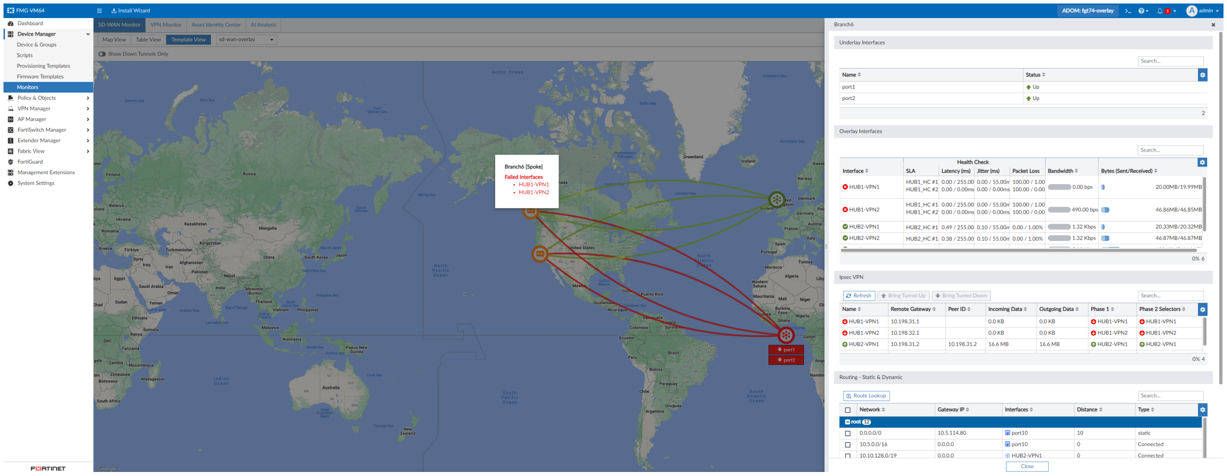
HUB6

Underlay Interfaces

Name	Status
port1	Up
port2	Up

2

6. Clicking on a spoke device opens a pane with the following information.
   - SD-WAN health check table.
   - Underlay status table.
   - IPsec VPN table.
   - Routing - Static & Dynamic table.



Branch6

Underlay Interfaces

Name	Status
port1	Up
port2	Up

2

Overlay Interfaces

Interface	Health Check				Bandwidth	Bytes (Sent/Received)
	SLA	Latency (ms)	Jitter (ms)	Packet Loss		
HUB1-VPN1	HUB1_HC #1 HUB1_HC #2	0.00 / 255.00 0.00 / 0.00ms	0.00 / 55.00m 0.00 / 0.00ms	100.00 / 1.00 100.00 / 0.00	0.00 bps	20.00MB/19.99MB
HUB1-VPN2	HUB1_HC #1 HUB1_HC #2	0.00 / 255.00 0.00 / 0.00ms	0.00 / 55.00m 0.00 / 0.00ms	100.00 / 1.00 100.00 / 0.00	490.00 bps	46.86MB/46.85MB
HUB2-VPN1	HUB2_HC #1	0.49 / 255.00	0.10 / 55.00m	0.00 / 1.00%	1.32 Kbps	20.33MB/20.32MB
HUB2-VPN2	HUB2_HC #1	0.38 / 255.00	0.10 / 55.00m	0.00 / 1.00%	1.32 Kbps	46.87MB/46.87MB

0% 6

Ipsec VPN

Name	Remote Gateway	Peer ID	Incoming Data	Outgoing Data	Phase 1	Phase 2 Selectors
HUB1-VPN1	10.198.31.1		0.0 KB	0.0 KB	HUB1-VPN1	HUB1-VPN1
HUB1-VPN2	10.198.32.1		0.0 KB	0.0 KB	HUB1-VPN2	HUB1-VPN2
HUB2-VPN1	10.198.31.2	10.198.31.2	16.6 MB	16.6 MB	HUB2-VPN1	HUB2-VPN1

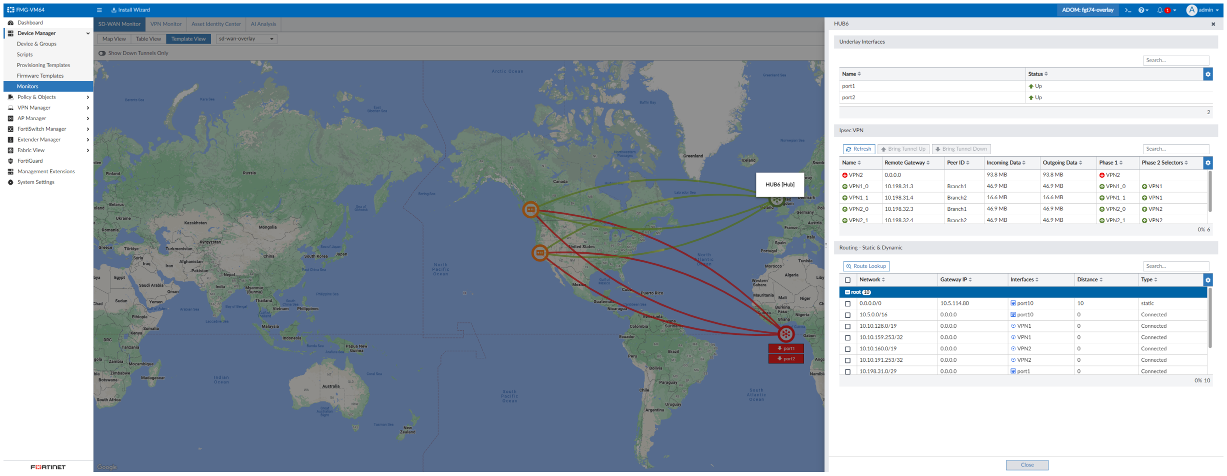
0% 4

Routing - Static & Dynamic

□	Network	Gateway IP	Interfaces	Distance	Type
root 12					
□	0.0.0.0/0	10.5.114.80	port10	10	static
□	10.5.0.0/16	0.0.0.0	port10	0	Connected
□	10.10.128.0/19	0.0.0.0	HUB2-VPN1	0	Connected

7. Clicking on a HUB device opens a pane with the following information:
   - Underlay Status table.
   - IPsec VPN table.

- Routing - Static & Dynamic table.



Underlay Interfaces

Name	Status
port1	Up
port2	Up

2

Ipsec VPN

Refresh | Bring Tunnel Up | Bring Tunnel Down

Name	Remote Gateway	Peer ID	Incoming Data	Outgoing Data	Phase 1	Phase 2 Selectors
VPN2	0.0.0.0		93.8 MB	93.8 MB	VPN2	
VPN1_0	10.198.31.3	Branch1	46.9 MB	46.9 MB	VPN1_0	VPN1
VPN1_1	10.198.31.4	Branch2	16.6 MB	16.6 MB	VPN1_1	VPN1
VPN2_0	10.198.32.3	Branch1	46.9 MB	46.9 MB	VPN2_0	VPN2
VPN2_1	10.198.32.4	Branch2	46.9 MB	46.9 MB	VPN2_1	VPN2

0% 6

Routing - Static & Dynamic

Route Lookup

□	Network	Gateway IP	Interfaces	Distance	Type
	root 10				
□	0.0.0.0/0	10.5.114.80	port10	10	static
□	10.5.0.0/16	0.0.0.0	port10	0	Connected
□	10.10.128.0/19	0.0.0.0	VPN1	0	Connected
□	10.10.159.253/32	0.0.0.0	VPN1	0	Connected
□	10.10.160.0/19	0.0.0.0	VPN2	0	Connected
□	10.10.191.253/32	0.0.0.0	VPN2	0	Connected
□	10.198.31.0/29	0.0.0.0	port1	0	Connected

0% 10

Improve client-side settings for SD-WAN network monitor - 7.4.1

Improvements have been made to the client-side settings of the SD-WAN network bandwidth monitoring service to increase the flexibility of the speed tests, and to optimize the settings to produce more accurate measurements. The changes include:

- Support UDP speed tests.
- Support multiple TCP connections to the server instead of a single connection.
- Measure the latency to speed test servers and select the server with the smallest latency to perform the test.
- Support the auto mode speed test, which selects either UDP or TCP testing automatically based on the latency threshold.

Summary of related CLI commands

To configure the speed test settings:

```
config system speed-test-setting
    set latency-threshold <integer>
    set multiple-tcp-stream <integer>
end
```

`latency-threshold <integer>`	Set the speed test threshold for the auto mode, in milliseconds (0 - 2000, default = 60). If the latency exceeds this threshold, the speed test will use the UDP protocol; otherwise, it will use the TCP protocol.
`multiple-tcp-stream <integer>`	Set the number of parallel client streams for the TCP protocol to run during the speed test (1 - 64, default = 4).

To run a manual interface speed test:

```
# execute speed-test <interface> <server> {Auto | TCP | UDP}
```

```
# diagnose netlink interface speed-test <interface> <server> {Auto | TCP | UDP}
```

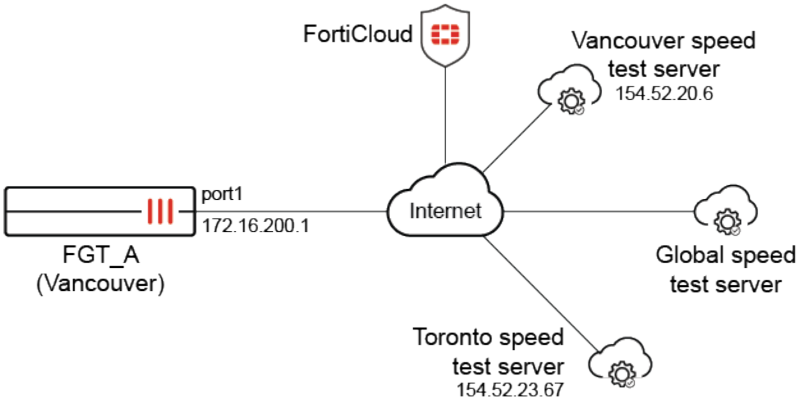
To configure the protocol mode for a speed test:

```
config system speed-test-schedule
    edit <interface>
        set mode {Auto | TCP | UDP}
    next
end
```

`Auto` is the default setting.

Examples

The following examples show various tests based on different modes (Auto, TCP, UDP), latency thresholds, and test servers. Some test protocols and servers are manually configured, while others are chosen by the FortiGate.



These examples assume the FortiGate is connected to the internet, has a valid SD-WAN Network Monitor license, and has downloaded the server list of speed tests from FortiCloud.

To download the server list of speed tests:

1. Download the server list from FortiCloud:

```
# execute speed-test-server download
Download completed.
```

2. Verify the list:

```
# execute speed-test-server list
...
FTNT_CA_Toronto valid
        Host: 154.52.23.67 5200 fortinet
        ...
FTNT_CA_Vancouver valid
        Host: 154.52.20.6 5200 fortinet
        ...
FTNT_Global valid
        Host: 154.52.6.95 5203 fortinet
        ...
```

Example 1: executing a speed test without specifying the interface, server, and mode

Geographically, the Vancouver server (154.52.20.6) has the smallest latency (around 7 ms) to FGT_A, so it will be automatically selected for the speed test because the latency 7 ms to 154.52.20.6 is less than the default `latency-threshold` of 60 ms. Meanwhile, four TCP connections will be initiated to perform the test since the default `multiple-tcp-stream` is 4.

To execute the speed test without specifying parameters:

1. Configure the speed test settings:

```
config system speed-test-setting
    set latency-threshold 60
    set multiple-tcp-stream 4
end
```

2. Execute a ping to the closest test server, 154.52.20.6, to learn the latency for the connection:

```
# execute ping 154.52.20.6
PING 154.52.20.6 (154.52.20.6): 56 data bytes
64 bytes from 154.52.20.6: icmp_seq=0 ttl=50 time=7.5 ms
64 bytes from 154.52.20.6: icmp_seq=1 ttl=50 time=7.2 ms
64 bytes from 154.52.20.6: icmp_seq=2 ttl=50 time=7.1 ms
64 bytes from 154.52.20.6: icmp_seq=3 ttl=50 time=7.1 ms
64 bytes from 154.52.20.6: icmp_seq=4 ttl=50 time=9.1 ms

--- 154.52.20.6 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 7.1/7.6/9.1 ms
```

3. Run the speed test with no parameters:

```
# execute speed-test
Initializing speed test.
current vdom=root
Run in uploading mode.
Connecting to host 154.52.20.6, port 5203
[  7] local 172.16.200.1 port 21219 connected to 154.52.20.6 port 5203
[  9] local 172.16.200.1 port 21220 connected to 154.52.20.6 port 5203
[ 11] local 172.16.200.1 port 21221 connected to 154.52.20.6 port 5203
[ 13] local 172.16.200.1 port 21222 connected to 154.52.20.6 port 5203
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  7]   0.00-1.00   sec  22.4 MBytes   188 Mbits/sec   17    140 KBytes
[  9]   0.00-1.00   sec  9.71 MBytes  81.4 Mbits/sec    6   73.5 KBytes
[ 11]   0.00-1.00   sec  18.5 MBytes   155 Mbits/sec   12    117 KBytes
[ 13]   0.00-1.00   sec  12.4 MBytes   104 Mbits/sec    7   87.7 KBytes
[SUM]   0.00-1.00   sec  63.1 MBytes   529 Mbits/sec   42
...
[ ID] Interval           Transfer     Bitrate         Retr
[  7]   0.00-5.00   sec  97.8 MBytes   164 Mbits/sec   45             sender
[  7]   0.00-5.02   sec  97.7 MBytes   163 Mbits/sec                  receiver
[  9]   0.00-5.00   sec  63.1 MBytes   106 Mbits/sec   14             sender
[  9]   0.00-5.02   sec  62.9 MBytes   105 Mbits/sec                  receiver
[ 11]   0.00-5.00   sec  80.1 MBytes   134 Mbits/sec   29             sender
[ 11]   0.00-5.02   sec  79.9 MBytes   134 Mbits/sec                  receiver
[ 13]   0.00-5.00   sec  80.3 MBytes   135 Mbits/sec   49             sender
[ 13]   0.00-5.02   sec  80.2 MBytes   134 Mbits/sec                  receiver
[SUM]   0.00-5.00   sec   321 MBytes   539 Mbits/sec  137             sender
[SUM]   0.00-5.02   sec   321 MBytes   536 Mbits/sec                  receiver

speed test Done.
Run in reverse downloading mode.
Connecting to host 154.52.20.6, port 5203
Reverse mode, remote host 154.52.20.6 is sending
[  7] local 172.16.200.1 port 21228 connected to 154.52.20.6 port 5203
[ 11] local 172.16.200.1 port 21229 connected to 154.52.20.6 port 5203
[ 15] local 172.16.200.1 port 21230 connected to 154.52.20.6 port 5203
[ 17] local 172.16.200.1 port 21231 connected to 154.52.20.6 port 5203
[ ID] Interval           Transfer     Bitrate
[  7]   0.00-1.00   sec  30.6 MBytes   256 Mbits/sec
[ 11]   0.00-1.00   sec  20.2 MBytes   170 Mbits/sec
[ 15]   0.00-1.00   sec  23.0 MBytes   193 Mbits/sec
[ 17]   0.00-1.00   sec  18.1 MBytes   152 Mbits/sec
[SUM]   0.00-1.00   sec  91.9 MBytes   771 Mbits/sec
...
[ ID] Interval           Transfer     Bitrate         Retr
[  7]   0.00-5.01   sec   101 MBytes   169 Mbits/sec  458             sender
[  7]   0.00-5.00   sec  97.4 MBytes   163 Mbits/sec                  receiver
[ 11]   0.00-5.01   sec  93.1 MBytes   156 Mbits/sec  266             sender
[ 11]   0.00-5.00   sec  91.8 MBytes   154 Mbits/sec                  receiver
[ 15]   0.00-5.01   sec  76.3 MBytes   128 Mbits/sec  201             sender
[ 15]   0.00-5.00   sec  74.7 MBytes   125 Mbits/sec                  receiver
[ 17]   0.00-5.01   sec  68.7 MBytes   115 Mbits/sec  219             sender
[ 17]   0.00-5.00   sec  66.8 MBytes   112 Mbits/sec                  receiver
[SUM]   0.00-5.01   sec   339 MBytes   568 Mbits/sec  1144             sender
[SUM]   0.00-5.00   sec   331 MBytes   555 Mbits/sec                  receiver

speed test Done.
```

The tested upload/download speed for port1 is 536 Mbps/555 Mbps when connecting to the closest server with four TCP connections.

Example 2: executing a speed test with a lower latency threshold setting

The `latency-threshold` setting is changed to 5 ms, which is less than the latency 7 ms to 154.52.20.6. When executing the speed test, one UDP connection will be initiated as expected.

To execute the speed test with a lower latency threshold setting:

1. Edit the speed test settings:

```
config system speed-test-setting
    set latency-threshold 5
end
```

2. Run the speed test:

```
# execute speed-test
Speed test quota for 7/19 is 4
current vdom=root
Run in uploading mode.
Connecting to host 154.52.20.6, port 5202
```

```
[  7] local 172.16.200.1 port 5315 connected to 154.52.20.6 port 5202
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  7]   0.00-1.00   sec   111 MBytes   931 Mbits/sec  80337
[  7]   1.00-2.00   sec   111 MBytes   932 Mbits/sec  80476
[  7]   2.00-3.00   sec   111 MBytes   932 Mbits/sec  80451
[  7]   3.00-4.00   sec   111 MBytes   932 Mbits/sec  80460
[  7]   4.00-5.00   sec   111 MBytes   934 Mbits/sec  80640
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  7]   0.00-5.00   sec   556 MBytes   932 Mbits/sec  0.000 ms  0/402364 (0%)  sender
[  7]   0.00-5.04   sec   550 MBytes   917 Mbits/sec  0.017 ms  3787/402339 (0.94%)
receiver

speed test Done.
Run in reverse downloading mode.
Connecting to host 154.52.20.6, port 5202
Reverse mode, remote host 154.52.20.6 is sending
[  7] local 172.16.200.1 port 19940 connected to 154.52.20.6 port 5202
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  7]   0.00-1.00   sec  72.4 MBytes   607 Mbits/sec  0.013 ms  59813/112240 (53%)
[  7]   1.00-2.00   sec  70.9 MBytes   595 Mbits/sec  0.015 ms  58130/109486 (53%)
[  7]   2.00-3.00   sec  69.2 MBytes   581 Mbits/sec  0.012 ms  60192/110329 (55%)
[  7]   3.00-4.00   sec  71.3 MBytes   598 Mbits/sec  0.012 ms  58107/109710 (53%)
[  7]   4.00-5.00   sec  71.1 MBytes   596 Mbits/sec  0.014 ms  58786/110260 (53%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  7]   0.00-5.04   sec   764 MBytes  1.27 Gbits/sec  0.000 ms  0/553023 (0%)  sender
[SUM]  0.0- 5.0 sec  2 datagrams received out-of-order
[  7]   0.00-5.00   sec   355 MBytes   595 Mbits/sec  0.014 ms  295028/552025 (53%)
receiver

speed test Done.
```

The tested upload/download speed for port1 is 917 Mbps/595 Mbps when connecting to the closest server with one UDP connection.

Example 3: executing a speed test with 10 TCP client streams

The `latency-threshold` setting is back to the 60 ms default, and the `multiple-tcp-stream` setting is changed to 10.

To execute the speed test with the default latency threshold and higher client stream value:

1. Edit the speed test settings:

```
config system speed-test-setting
    set latency-threshold 60
    set multiple-tcp-stream 10
end
```

2. Run the speed test:

```
# execute speed-test
Speed test quota for 7/19 is 3
current vdom=root
Run in uploading mode.
```

```
Connecting to host 154.52.20.6, port 5203
[  7] local 172.16.200.1 port 22373 connected to 154.52.20.6 port 5203
[  9] local 172.16.200.1 port 22374 connected to 154.52.20.6 port 5203
[ 11] local 172.16.200.1 port 22375 connected to 154.52.20.6 port 5203
[ 13] local 172.16.200.1 port 22376 connected to 154.52.20.6 port 5203
[ 15] local 172.16.200.1 port 22377 connected to 154.52.20.6 port 5203
[ 17] local 172.16.200.1 port 22378 connected to 154.52.20.6 port 5203
[ 19] local 172.16.200.1 port 22379 connected to 154.52.20.6 port 5203
[ 21] local 172.16.200.1 port 22380 connected to 154.52.20.6 port 5203
[ 23] local 172.16.200.1 port 22381 connected to 154.52.20.6 port 5203
[ 25] local 172.16.200.1 port 22382 connected to 154.52.20.6 port 5203
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  7]   0.00-1.00   sec  15.1 MBytes   127 Mbits/sec   14   72.1 KBytes
[  9]   0.00-1.00   sec  8.42 MBytes  70.6 Mbits/sec    9   43.8 KBytes
[ 11]   0.00-1.00   sec  11.9 MBytes  99.8 Mbits/sec   11   82.0 KBytes
[ 13]   0.00-1.00   sec  8.12 MBytes  68.0 Mbits/sec   10   55.1 KBytes
[ 15]   0.00-1.00   sec  5.49 MBytes  46.1 Mbits/sec   11   32.5 KBytes
[ 17]   0.00-1.00   sec  5.77 MBytes  48.3 Mbits/sec    7   59.4 KBytes
[ 19]   0.00-1.00   sec  17.8 MBytes   149 Mbits/sec   16    133 KBytes
[ 21]   0.00-1.00   sec  9.52 MBytes  79.8 Mbits/sec    7   67.9 KBytes
[ 23]   0.00-1.00   sec  4.84 MBytes  40.6 Mbits/sec    7   35.4 KBytes
[ 25]   0.00-1.00   sec  7.92 MBytes  66.4 Mbits/sec    9   79.2 KBytes
[SUM]   0.00-1.00   sec  94.9 MBytes   796 Mbits/sec  101
...
[ ID] Interval           Transfer     Bitrate         Retr
[  7]   0.00-5.00   sec  52.7 MBytes  88.3 Mbits/sec   34             sender
[  7]   0.00-5.01   sec  52.5 MBytes  88.0 Mbits/sec                  receiver
[  9]   0.00-5.00   sec  40.8 MBytes  68.5 Mbits/sec   22             sender
[  9]   0.00-5.01   sec  40.7 MBytes  68.2 Mbits/sec                  receiver
[ 11]   0.00-5.00   sec  42.8 MBytes  71.7 Mbits/sec   26             sender
[ 11]   0.00-5.01   sec  42.7 MBytes  71.5 Mbits/sec                  receiver
[ 13]   0.00-5.00   sec  34.8 MBytes  58.4 Mbits/sec   27             sender
[ 13]   0.00-5.01   sec  34.7 MBytes  58.1 Mbits/sec                  receiver
[ 15]   0.00-5.00   sec  38.7 MBytes  64.8 Mbits/sec   23             sender
[ 15]   0.00-5.01   sec  38.6 MBytes  64.6 Mbits/sec                  receiver
[ 17]   0.00-5.00   sec  35.7 MBytes  59.9 Mbits/sec   22             sender
[ 17]   0.00-5.01   sec  35.7 MBytes  59.8 Mbits/sec                  receiver
[ 19]   0.00-5.00   sec  58.2 MBytes  97.5 Mbits/sec   39             sender
[ 19]   0.00-5.01   sec  57.9 MBytes  97.0 Mbits/sec                  receiver
[ 21]   0.00-5.00   sec  34.2 MBytes  57.4 Mbits/sec   29             sender
[ 21]   0.00-5.01   sec  34.1 MBytes  57.2 Mbits/sec                  receiver
[ 23]   0.00-5.00   sec  29.6 MBytes  49.7 Mbits/sec   26             sender
[ 23]   0.00-5.01   sec  29.6 MBytes  49.5 Mbits/sec                  receiver
[ 25]   0.00-5.00   sec  54.6 MBytes  91.5 Mbits/sec   28             sender
[ 25]   0.00-5.01   sec  54.5 MBytes  91.3 Mbits/sec                  receiver
[SUM]   0.00-5.00   sec   422 MBytes   708 Mbits/sec  276             sender
[SUM]   0.00-5.01   sec   421 MBytes   705 Mbits/sec                  receiver

speed test Done.
Run in reverse downloading mode.
Connecting to host 154.52.20.6, port 5203
Reverse mode, remote host 154.52.20.6 is sending
[  7] local 172.16.200.1 port 22384 connected to 154.52.20.6 port 5203
[ 11] local 172.16.200.1 port 22385 connected to 154.52.20.6 port 5203
[ 15] local 172.16.200.1 port 22386 connected to 154.52.20.6 port 5203
[ 19] local 172.16.200.1 port 22387 connected to 154.52.20.6 port 5203
[ 23] local 172.16.200.1 port 22388 connected to 154.52.20.6 port 5203
[ 27] local 172.16.200.1 port 22389 connected to 154.52.20.6 port 5203
[ 29] local 172.16.200.1 port 22390 connected to 154.52.20.6 port 5203
[ 31] local 172.16.200.1 port 22391 connected to 154.52.20.6 port 5203
[ 33] local 172.16.200.1 port 22392 connected to 154.52.20.6 port 5203
[ 35] local 172.16.200.1 port 22393 connected to 154.52.20.6 port 5203
[ ID] Interval           Transfer     Bitrate
[  7]   0.00-1.00   sec  11.5 MBytes  96.7 Mbits/sec
[ 11]   0.00-1.00   sec  7.97 MBytes  66.9 Mbits/sec
[ 15]   0.00-1.00   sec  6.19 MBytes  52.0 Mbits/sec
[ 19]   0.00-1.00   sec  8.27 MBytes  69.4 Mbits/sec
[ 23]   0.00-1.00   sec  8.34 MBytes  69.9 Mbits/sec
[ 27]   0.00-1.00   sec  5.85 MBytes  49.0 Mbits/sec
[ 29]   0.00-1.00   sec  7.64 MBytes  64.1 Mbits/sec
[ 31]   0.00-1.00   sec  5.61 MBytes  47.0 Mbits/sec
[ 33]   0.00-1.00   sec  6.95 MBytes  58.3 Mbits/sec
[ 35]   0.00-1.00   sec  6.43 MBytes  53.9 Mbits/sec
[SUM]   0.00-1.00   sec  74.8 MBytes   627 Mbits/sec
...
[ ID] Interval           Transfer     Bitrate         Retr
[  7]   0.00-5.01   sec  39.4 MBytes  65.9 Mbits/sec  197             sender
[  7]   0.00-5.00   sec  37.6 MBytes  63.0 Mbits/sec                  receiver
[ 11]   0.00-5.01   sec  49.0 MBytes  82.1 Mbits/sec  216             sender
[ 11]   0.00-5.00   sec  48.1 MBytes  80.8 Mbits/sec                  receiver
[ 15]   0.00-5.01   sec  27.4 MBytes  45.9 Mbits/sec  206             sender
[ 15]   0.00-5.00   sec  26.4 MBytes  44.3 Mbits/sec                  receiver
[ 19]   0.00-5.01   sec  42.6 MBytes  71.3 Mbits/sec  158             sender
[ 19]   0.00-5.00   sec  42.1 MBytes  70.6 Mbits/sec                  receiver
[ 23]   0.00-5.01   sec  37.1 MBytes  62.2 Mbits/sec  174             sender
[ 23]   0.00-5.00   sec  36.6 MBytes  61.4 Mbits/sec                  receiver
[ 27]   0.00-5.01   sec  34.6 MBytes  58.0 Mbits/sec  161             sender
[ 27]   0.00-5.00   sec  34.1 MBytes  57.2 Mbits/sec                  receiver
[ 29]   0.00-5.01   sec  40.2 MBytes  67.4 Mbits/sec  135             sender
[ 29]   0.00-5.00   sec  39.6 MBytes  66.5 Mbits/sec                  receiver
[ 31]   0.00-5.01   sec  40.9 MBytes  68.5 Mbits/sec  172             sender
[ 31]   0.00-5.00   sec  40.4 MBytes  67.8 Mbits/sec                  receiver
[ 33]   0.00-5.01   sec  35.4 MBytes  59.3 Mbits/sec  164             sender
[ 33]   0.00-5.00   sec  34.9 MBytes  58.5 Mbits/sec                  receiver
[ 35]   0.00-5.01   sec  37.2 MBytes  62.3 Mbits/sec  148             sender
[ 35]   0.00-5.00   sec  36.7 MBytes  61.5 Mbits/sec                  receiver
[SUM]   0.00-5.01   sec   384 MBytes   643 Mbits/sec  1731             sender
[SUM]   0.00-5.00   sec   377 MBytes   632 Mbits/sec                  receiver

speed test Done.
```

The tested upload/download speed for port1 is 705 Mbps/632 Mbps when connecting to the closest server with 10 TCP connections.

Example 4: executing a speed test by specifying the interface, server, and UDP mode

The speed test will test the Toronto server using UDP mode on port1.

To execute the speed test:

```
# execute speed-test port1 FTNT_CA_Toronto UDP
Speed test quota for 7/19 is 1
bind to local ip 172.16.200.1
current vdom=root
Run in uploading mode.
Connecting to host 154.52.23.67, port 5201
[  7] local 172.16.200.1 port 10860 connected to 154.52.23.67 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  7]   0.00-1.00   sec   112 MBytes   936 Mbits/sec  80759
[  7]   1.00-2.00   sec   112 MBytes   937 Mbits/sec  80886
[  7]   2.00-3.00   sec   112 MBytes   937 Mbits/sec  80903
[  7]   3.00-4.00   sec   111 MBytes   935 Mbits/sec  80677
[  7]   4.00-5.00   sec   111 MBytes   934 Mbits/sec  80600
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  7]   0.00-5.00   sec   558 MBytes   936 Mbits/sec  0.000 ms  0/403825 (0%)  sender
[  7]   0.00-5.09   sec   552 MBytes   908 Mbits/sec  0.013 ms  4435/403815 (1.1%)  receiver

speed test Done.
Run in reverse downloading mode.
Connecting to host 154.52.23.67, port 5201
Reverse mode, remote host 154.52.23.67 is sending
[  7] local 172.16.200.1 port 15370 connected to 154.52.23.67 port 5201
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  7]   0.00-1.00   sec  58.8 MBytes   493 Mbits/sec  0.017 ms  60888/103447 (59%)
[  7]   1.00-2.00   sec  58.3 MBytes   489 Mbits/sec  0.012 ms  93083/135310 (69%)
[  7]   2.00-3.00   sec  59.4 MBytes   499 Mbits/sec  0.017 ms  95066/138106 (69%)
[  7]   3.00-4.00   sec  54.0 MBytes   453 Mbits/sec  0.024 ms  97539/136672 (71%)
[  7]   4.00-5.00   sec  58.6 MBytes   491 Mbits/sec  0.015 ms  93797/136213 (69%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  7]   0.00-5.10   sec   908 MBytes  1.49 Gbits/sec  0.000 ms  0/657629 (0%)  sender
[  7]   0.00-5.00   sec   289 MBytes   485 Mbits/sec  0.015 ms  440373/649748 (68%)
receiver

speed test Done.
```

The tested upload/download speed for port1 is 908 Mbps/485 Mbps when connecting to the Toronto server with one UDP connection.

Example 5: executing a speed test by specifying the interface, server, and auto mode

The speed test will test the Toronto server using auto mode on port1. Since the latency to the Toronto server is less than 60 ms, 10 TCP connections are initiated.

To execute the speed test:

```
# execute speed-test port1 FTNT_CA_Toronto Auto
Speed test quota for 7/19 is 8
bind to local ip 172.16.200.1
current vdom=root
Run in uploading mode.
Connecting to host 154.52.23.67, port 5200
[  7] local 172.16.200.1 port 4333 connected to 154.52.23.67 port 5200
[  9] local 172.16.200.1 port 4334 connected to 154.52.23.67 port 5200
[ 11] local 172.16.200.1 port 4335 connected to 154.52.23.67 port 5200
[ 13] local 172.16.200.1 port 4336 connected to 154.52.23.67 port 5200
[ 15] local 172.16.200.1 port 4337 connected to 154.52.23.67 port 5200
[ 17] local 172.16.200.1 port 4338 connected to 154.52.23.67 port 5200
[ 19] local 172.16.200.1 port 4339 connected to 154.52.23.67 port 5200
[ 21] local 172.16.200.1 port 4340 connected to 154.52.23.67 port 5200
[ 23] local 172.16.200.1 port 4341 connected to 154.52.23.67 port 5200
[ 25] local 172.16.200.1 port 4342 connected to 154.52.23.67 port 5200
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  7]   0.00-1.00   sec  1.61 MBytes  13.5 Mbits/sec    1    264 KBytes
[  9]   0.00-1.00   sec  1.06 MBytes  8.90 Mbits/sec    0    160 KBytes
[ 11]   0.00-1.00   sec  1.35 MBytes  11.3 Mbits/sec    0    184 KBytes
[ 13]   0.00-1.00   sec  1.46 MBytes  12.2 Mbits/sec    0    222 KBytes
[ 15]   0.00-1.00   sec  1.32 MBytes  11.1 Mbits/sec    0    182 KBytes
[ 17]   0.00-1.00   sec  1.79 MBytes  15.0 Mbits/sec    0    263 KBytes
[ 19]   0.00-1.00   sec   912 KBytes  7.46 Mbits/sec    0   97.6 KBytes
[ 21]   0.00-1.00   sec  1.47 MBytes  12.3 Mbits/sec    0    188 KBytes
[ 23]   0.00-1.00   sec  1.04 MBytes  8.75 Mbits/sec    0    175 KBytes
[ 25]   0.00-1.00   sec   929 KBytes  7.60 Mbits/sec    0   94.7 KBytes
[SUM]   0.00-1.00   sec  12.9 MBytes   108 Mbits/sec    1
...
[ ID] Interval           Transfer     Bitrate         Retr
[  7]   0.00-5.00   sec  28.1 MBytes  47.1 Mbits/sec    8             sender
[  7]   0.00-5.05   sec  27.5 MBytes  45.7 Mbits/sec                  receiver
[  9]   0.00-5.00   sec  11.8 MBytes  19.8 Mbits/sec   10             sender
[  9]   0.00-5.05   sec  11.1 MBytes  18.5 Mbits/sec                  receiver
[ 11]   0.00-5.00   sec  40.5 MBytes  68.0 Mbits/sec   11             sender
[ 11]   0.00-5.05   sec  40.1 MBytes  66.7 Mbits/sec                  receiver
[ 13]   0.00-5.00   sec  18.0 MBytes  30.2 Mbits/sec    6             sender
[ 13]   0.00-5.05   sec  17.6 MBytes  29.2 Mbits/sec                  receiver
[ 15]   0.00-5.00   sec  38.8 MBytes  65.2 Mbits/sec    1             sender
[ 15]   0.00-5.05   sec  38.8 MBytes  64.4 Mbits/sec                  receiver
[ 17]   0.00-5.00   sec  15.0 MBytes  25.2 Mbits/sec   10             sender
[ 17]   0.00-5.05   sec  14.8 MBytes  24.5 Mbits/sec                  receiver
[ 19]   0.00-5.00   sec  20.5 MBytes  34.4 Mbits/sec    1             sender
[ 19]   0.00-5.05   sec  20.3 MBytes  33.7 Mbits/sec                  receiver
[ 21]   0.00-5.00   sec  13.9 MBytes  23.2 Mbits/sec   12             sender
[ 21]   0.00-5.05   sec  13.2 MBytes  21.9 Mbits/sec                  receiver
[ 23]   0.00-5.00   sec  7.59 MBytes  12.7 Mbits/sec   13             sender
[ 23]   0.00-5.05   sec  7.37 MBytes  12.2 Mbits/sec                  receiver
[ 25]   0.00-5.00   sec  17.7 MBytes  29.7 Mbits/sec   10             sender
[ 25]   0.00-5.05   sec  17.4 MBytes  28.9 Mbits/sec                  receiver
[SUM]   0.00-5.00   sec   212 MBytes   355 Mbits/sec   82             sender
[SUM]   0.00-5.05   sec   208 MBytes   346 Mbits/sec                  receiver

speed test Done.
```

```
Run in reverse downloading mode.
Connecting to host 154.52.23.67, port 5200
Reverse mode, remote host 154.52.23.67 is sending
[  7] local 172.16.200.1 port 4344 connected to 154.52.23.67 port 5200
[ 11] local 172.16.200.1 port 4345 connected to 154.52.23.67 port 5200
[ 15] local 172.16.200.1 port 4346 connected to 154.52.23.67 port 5200
[ 19] local 172.16.200.1 port 4347 connected to 154.52.23.67 port 5200
[ 23] local 172.16.200.1 port 4348 connected to 154.52.23.67 port 5200
[ 27] local 172.16.200.1 port 4349 connected to 154.52.23.67 port 5200
[ 29] local 172.16.200.1 port 4350 connected to 154.52.23.67 port 5200
[ 31] local 172.16.200.1 port 4351 connected to 154.52.23.67 port 5200
[ 33] local 172.16.200.1 port 4352 connected to 154.52.23.67 port 5200
[ 35] local 172.16.200.1 port 4353 connected to 154.52.23.67 port 5200
[ ID] Interval           Transfer     Bitrate
[  7]   0.00-1.00   sec  2.31 MBytes  19.3 Mbits/sec
[ 11]   0.00-1.00   sec  2.70 MBytes  22.6 Mbits/sec
[ 15]   0.00-1.00   sec  1.80 MBytes  15.1 Mbits/sec
[ 19]   0.00-1.00   sec  2.33 MBytes  19.5 Mbits/sec
[ 23]   0.00-1.00   sec  1.30 MBytes  10.9 Mbits/sec
[ 27]   0.00-1.00   sec  1.55 MBytes  13.0 Mbits/sec
[ 29]   0.00-1.00   sec  3.65 MBytes  30.5 Mbits/sec
[ 31]   0.00-1.00   sec  1.35 MBytes  11.3 Mbits/sec
[ 33]   0.00-1.00   sec  3.26 MBytes  27.3 Mbits/sec
[ 35]   0.00-1.00   sec  2.85 MBytes  23.8 Mbits/sec
[SUM]   0.00-1.00   sec  23.1 MBytes   193 Mbits/sec
...
[ ID] Interval           Transfer     Bitrate         Retr
[  7]   0.00-5.06   sec  16.2 MBytes  26.9 Mbits/sec   33             sender
[  7]   0.00-5.00   sec  14.6 MBytes  24.5 Mbits/sec                  receiver
[ 11]   0.00-5.06   sec  13.9 MBytes  23.0 Mbits/sec   64             sender
[ 11]   0.00-5.00   sec  12.9 MBytes  21.6 Mbits/sec                  receiver
[ 15]   0.00-5.06   sec  8.61 MBytes  14.3 Mbits/sec   75             sender
[ 15]   0.00-5.00   sec  7.63 MBytes  12.8 Mbits/sec                  receiver
[ 19]   0.00-5.06   sec  11.9 MBytes  19.7 Mbits/sec   65             sender
[ 19]   0.00-5.00   sec  10.8 MBytes  18.2 Mbits/sec                  receiver
[ 23]   0.00-5.06   sec  7.37 MBytes  12.2 Mbits/sec   13             sender
[ 23]   0.00-5.00   sec  6.77 MBytes  11.4 Mbits/sec                  receiver
[ 27]   0.00-5.06   sec  7.44 MBytes  12.3 Mbits/sec   86             sender
[ 27]   0.00-5.00   sec  6.47 MBytes  10.8 Mbits/sec                  receiver
[ 29]   0.00-5.06   sec  19.0 MBytes  31.5 Mbits/sec   27             sender
[ 29]   0.00-5.00   sec  17.7 MBytes  29.6 Mbits/sec                  receiver
[ 31]   0.00-5.06   sec  7.11 MBytes  11.8 Mbits/sec   51             sender
[ 31]   0.00-5.00   sec  6.43 MBytes  10.8 Mbits/sec                  receiver
[ 33]   0.00-5.06   sec  21.5 MBytes  35.7 Mbits/sec   23             sender
[ 33]   0.00-5.00   sec  20.4 MBytes  34.2 Mbits/sec                  receiver
[ 35]   0.00-5.06   sec  18.4 MBytes  30.5 Mbits/sec   48             sender
[ 35]   0.00-5.00   sec  17.0 MBytes  28.6 Mbits/sec                  receiver
[SUM]   0.00-5.06   sec   131 MBytes   218 Mbits/sec  485             sender
[SUM]   0.00-5.00   sec   121 MBytes   202 Mbits/sec                  receiver

speed test Done.
```

The tested upload/download speed for port1 is 346 Mbps/202 Mbps when connecting to the Toronto server with 10 TCP connections.

Example 6: executing the speed test with diagnose netlink interface speed-test

After running this diagnose command, the results are recorded in the interface settings for reference as `measured-upstream-bandwidth` and `measured-downstream-bandwidth`.

To execute the speed test:

```
# diagnose netlink interface speed-test port1 FTNT_CA_Vancouver TCP
speed-test test ID is b0066
...
```

To view the interface settings:

```
show system interface port1
config system interface
    edit "port1"
        ...
        set measured-upstream-bandwidth 735682
        set measured-downstream-bandwidth 746573
        set bandwidth-measure-time 1689811319
        ...
    next
end
```

Example 7: executing the speed test according to the schedule

After running the speed test, the results are recorded in the interface settings for reference as `measured-upstream-bandwidth` and `measured-downstream-bandwidth`.

To execute the speed test according to the schedule:

1. Configure the speed test schedule:

   ```
   config system speed-test-schedule
       edit "port1"
           set mode TCP
           set schedules "speedtest_recurring"
       next
   end
   ```

2. Configure the recurring schedule:

   ```
   config firewall schedule recurring
       edit "speedtest_recurring"
           set start 17:07
           set day sunday monday tuesday wednesday thursday friday saturday
       next
   end
   ```

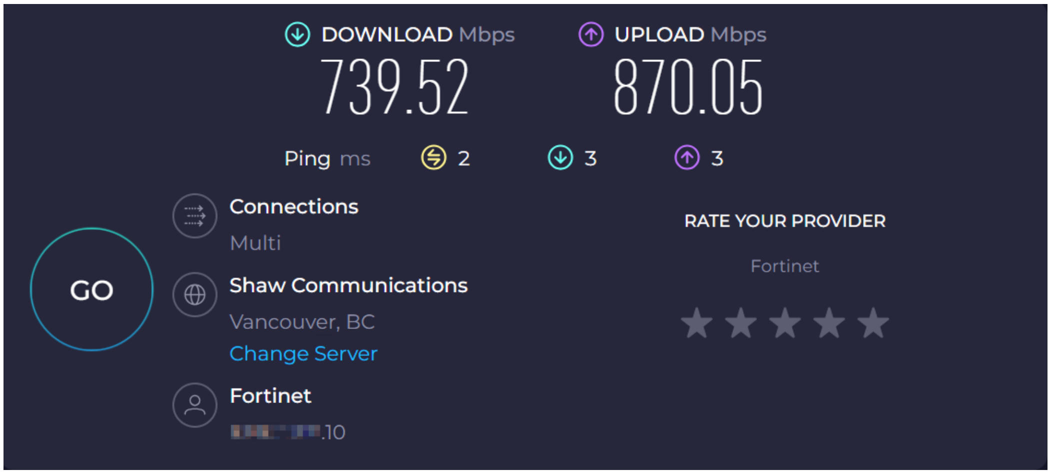
   The speed test will be initiated at 17:07 based on 10 TCP connections. The results will be recorded in port1's interface settings.

3. Verify the interface settings:

```
show system interface port1
config system interface
    edit "port1"
        ...
        set measured-upstream-bandwidth 715636
        set measured-downstream-bandwidth 819682
        set bandwidth-measure-time 1689811759
        ...
    next
end
```

Example 8: executing multiple speed tests with TCP and UDP connections

A speed test is executed to the closest server using 64 TCP connections and another speed test is executed using one UDP connection. The results can be checked with a third-party platform (such as Ookla), which returns comparable results.



To execute multiple speed tests with TCP and UDP connections:

1. Edit the speed test settings:

```
config system speed-test-setting
    set multiple-tcp-stream 64
end
```

2. Run the TCP speed test:

```
# execute speed-test port1 FTNT_CA_Vancouver TCP
...
Run in uploading mode.
...
[SUM]   0.00-5.00   sec   559 MBytes   938 Mbits/sec  2165             sender
[SUM]   0.00-5.01   sec   558 MBytes   933 Mbits/sec                  receiver

speed test Done.
Run in reverse downloading mode.
...
[SUM]   0.00-5.01   sec   505 MBytes   846 Mbits/sec  9329             sender
[SUM]   0.00-5.00   sec   491 MBytes   823 Mbits/sec                  receiver
```

3. Run the UDP speed test:

```
# execute speed-test port1 FTNT_CA_Vancouver UDP
...
Run in uploading mode.
...
[  7]   0.00-5.00   sec   556 MBytes   933 Mbits/sec  0.000 ms  0/402727 (0%)  sender
[  7]   0.00-5.04   sec   556 MBytes   925 Mbits/sec  0.020 ms  393/402717 (0.098%)
receiver
...
Run in reverse downloading mode.
...
[  7]   0.00-5.04   sec   869 MBytes  1.45 Gbits/sec  0.000 ms  0/629383 (0%)  sender
[SUM]  0.0- 5.0 sec  2 datagrams received out-of-order
[  7]   0.00-5.00   sec   489 MBytes   821 Mbits/sec  0.005 ms  274103/628393 (44%)
receiver

speed test Done.
```

Multiple interface monitoring for IPsec - 7.4.1

IPsec can monitor multiple interfaces per tunnel, and activate a backup link only when all of the primary links are down. This can be useful if you have multiple WAN links and want to optimize your WAN link selection and performance while limiting the use of more expensive and bandwidth intensive interfaces, like 5G or LTE.

In cases where multiple primary overlays are deployed and the backup overlay is on an LTE connection, avoiding IPsec keep alive messages, BGP hellos, and SD-WAN health checks on the backup connection is required when the primary overlays are working. The backup overlay can monitor all of the primary overlays, and is not activated until the number of unhealthy primary overlays equals or surpasses the predefined threshold.

```
config vpn ipsec phase1-interface
    edit <phase-1 name>
        set monitor <overlay> <overlay> ... <overlay>
        set monitor-min <integer>
    next
end
```

`monitor`	The IPsec interfaces to monitor.
`monitor-min`	The minimum number of monitored interfaces that must become degraded before this interface is activated (0 = all interfaces, default = 0).

In this example, four primary overlays are configured, T1 - T4, on fixed broadband connections and one backup overlay, T5, is configured on an LTE connection.

The backup overlay stays down as long as the primary overlays are working normally. When all four of the primary overlays go down, the backup overlay is activated and used to forward traffic. If any of the primary overlays recover, then the backup overlay goes down.

SD-WAN can also be configured to steer traffic.

To configure the overlays:

1. Configure the VPN remote gateways:

```
config vpn ipsec phase1-interface
    edit "T1"
        set interface "dmz"
        set ike-version 2
        set peertype any
        set net-device disable
        set proposal aes128-sha256
        set remote-gw 172.16.208.2
        set psksecret **********
    next
    edit "T2"
        set interface "agg1"
        set ike-version 2
        set peertype any
        set net-device disable
        set proposal aes128-sha256
        set remote-gw 172.16.203.2
        set psksecret **********
    next
    edit "T3"
        set interface "vlan100"
        set ike-version 2
        set peertype any
        set net-device disable
        set proposal aes128-sha256
        set remote-gw 172.16.206.2
        set psksecret **********
    next
    edit "T4"
        set interface "port15"
        set ike-version 2
        set peertype any
        set net-device disable
        set proposal aes128-sha256
        set remote-gw 172.16.209.2
        set psksecret **********
    next
    edit "T5"
        set interface "vlan200"
        set ike-version 2
        set peertype any
        set monitor "T1" "T2" "T3" "T4"
        set monitor-min 4
        set net-device disable
        set proposal aes128-sha256
        set remote-gw 172.16.210.2
        set psksecret **********
    next
end
```

2. Configure the VPN tunnels:

```
config vpn ipsec phase2-interface
    edit "T1_P2"
        set phase1name "T1"
        set proposal aes256-sha256
        set auto-negotiate enable
    next
    edit "T2_P2"
        set phase1name "T2"
        set proposal aes256-sha256
        set auto-negotiate enable
    next
    edit "T3_P2"
        set phase1name "T3"
        set proposal aes256-sha256
        set auto-negotiate enable
    next
    edit "T4_P2"
        set phase1name "T4"
        set proposal aes256-sha256
        set auto-negotiate enable
    next
    edit "T5_P2"
        set phase1name "T5"
        set proposal aes256-sha256
        set auto-negotiate enable
    next
end
```

3. Configure the interfaces:

```
config system interface
    edit "T1"
        set vdom "root"
        set ip 100.1.1.1 255.255.255.255
        set allowaccess ping
        set type tunnel
        set remote-ip 100.1.1.2 255.255.255.0
        set snmp-index 113
        set interface "dmz"
    next
    edit "T2"
        set vdom "root"
        set ip 100.1.2.1 255.255.255.255
        set allowaccess ping
        set type tunnel
        set remote-ip 100.1.2.2 255.255.255.0
        set snmp-index 114
        set interface "agg1"
    next
    edit "T3"
        set vdom "root"
        set ip 100.1.3.1 255.255.255.255
        set allowaccess ping
        set type tunnel
        set remote-ip 100.1.3.2 255.255.255.0
        set snmp-index 115
        set interface "vlan100"
    next
    edit "T4"
        set vdom "root"
        set ip 100.1.4.1 255.255.255.255
        set allowaccess ping
        set type tunnel
        set remote-ip 100.1.4.2 255.255.255.0
        set snmp-index 65
        set interface "port15"
    next
    edit "T5"
        set vdom "root"
        set ip 100.1.5.1 255.255.255.255
        set allowaccess ping
        set type tunnel
        set remote-ip 100.1.5.2 255.255.255.0
        set snmp-index 117
        set interface "vlan200"
    next
end
```

4. Check the IPsec tunnel summary:

```
# get vpn ipsec tunnel summary
'T2' 172.16.203.2:0  selectors(total,up): 1/1  rx(pkt,err): 0/0  tx(pkt,err): 0/4
'T3' 172.16.206.2:0  selectors(total,up): 1/1  rx(pkt,err): 0/0  tx(pkt,err): 0/4
'T4' 172.16.209.2:0  selectors(total,up): 1/1  rx(pkt,err): 0/0  tx(pkt,err): 0/4
'T5' 172.16.210.2:0  selectors(total,up): 1/0  rx(pkt,err): 0/0  tx(pkt,err): 0/4
'T1' 172.16.208.2:0  selectors(total,up): 1/1  rx(pkt,err): 0/0  tx(pkt,err): 0/4
```

The backup overlay, T5, is down.

To configure steering traffic with SD-WAN:

1. Configure the SD-WAN:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "T1"
        next
        edit 2
            set interface "T2"
        next
        edit 3
            set interface "T3"
        next
        edit 4
            set interface "T4"
        next
        edit 5
            set interface "T5"
        next
    end
    config service
        edit 1
            set name "1"
            set load-balance enable
            set dst "all"
            set src "172.16.205.0"
            set priority-members 1 2 3 4 5
        next
    end
end
```

2. Configure a static route:

```
config router static
    edit 5
        set dst 8.0.0.0 255.0.0.0
        set distance 1
        set sdwan-zone "virtual-wan-link"
    next
end
```

3. Check the routing table:

```
# get router info routing-table static
Routing table for VRF=0
S       8.0.0.0/8 [1/0] via T2 tunnel 172.16.203.2, [1/0]
                  [1/0] via T3 tunnel 172.16.206.2, [1/0]
                  [1/0] via T1 tunnel 172.16.208.2, [1/0]
                  [1/0] via T4 tunnel 172.16.209.2, [1/0]
```

Check the results:

- When both the T1 and T2 connections are down, T5 stays down as well, and traffic is load-balanced on T3 and T4 by the SD-WAN configuration:

```
 # get vpn ipsec tunnel summary
'T2' 172.16.203.2:0  selectors(total,up): 1/0  rx(pkt,err): 0/0  tx(pkt,err): 0/0
'T3' 172.16.206.2:0  selectors(total,up): 1/1  rx(pkt,err): 0/0  tx(pkt,err): 0/0
'T4' 172.16.209.2:0  selectors(total,up): 1/1  rx(pkt,err): 0/0  tx(pkt,err): 0/4
'T5' 172.16.210.2:0  selectors(total,up): 1/0  rx(pkt,err): 0/0  tx(pkt,err): 0/4
'T1' 172.16.208.2:0  selectors(total,up): 1/0  rx(pkt,err): 0/0  tx(pkt,err): 0/0
```

```
# get router info routing-table static
Routing table for VRF=0
S       8.0.0.0/8 [1/0] via T3 tunnel 172.16.206.2, [1/0]
                  [1/0] via T4 tunnel 172.16.209.2, [1/0]
```

Traffic is load-balanced between the remaining tunnels:

```
# diagnose sniffer packet any 'host 8.8.8.8' 4
interfaces=[any]
filters=[host 8.8.8.8]
3.027055 port5 in 172.16.205.100 -> 8.8.8.8: icmp: echo request
3.027154 T4 out 172.16.205.100 -> 8.8.8.8: icmp: echo request
3.031434 T4 in 8.8.8.8 -> 172.16.205.100: icmp: echo reply
3.031485 port5 out 8.8.8.8 -> 172.16.205.100: icmp: echo reply
```

```
3.612818 port5 in 172.16.205.100 -> 8.8.8.8: icmp: echo request
3.612902 T3 out 172.16.205.100 -> 8.8.8.8: icmp: echo request
3.617107 T3 in 8.8.8.8 -> 172.16.205.100: icmp: echo reply
3.617159 port5 out 8.8.8.8 -> 172.16.205.100: icmp: echo reply
4.168845 port5 in 172.16.205.100 -> 8.8.8.8: icmp: echo request
4.168907 T4 out 172.16.205.100 -> 8.8.8.8: icmp: echo request
4.173150 T4 in 8.8.8.8 -> 172.16.205.100: icmp: echo reply
4.173174 port5 out 8.8.8.8 -> 172.16.205.100: icmp: echo reply
4.710907 port5 in 172.16.205.100 -> 8.8.8.8: icmp: echo request
4.710991 T3 out 172.16.205.100 -> 8.8.8.8: icmp: echo request
4.715933 T3 in 8.8.8.8 -> 172.16.205.100: icmp: echo reply
4.715958 port5 out 8.8.8.8 -> 172.16.205.100: icmp: echo reply
```

- When all of the primary overlays are down, T5 is activated and used for traffic

```
# get vpn ipsec tunnel summary
'T2' 172.16.203.2:0  selectors(total,up): 1/0  rx(pkt,err): 0/0  tx(pkt,err): 0/0
'T3' 172.16.206.2:0  selectors(total,up): 1/0  rx(pkt,err): 0/0  tx(pkt,err): 0/0
'T4' 172.16.209.2:0  selectors(total,up): 1/0  rx(pkt,err): 0/0  tx(pkt,err): 0/0
'T5' 172.16.210.2:0  selectors(total,up): 1/1  rx(pkt,err): 0/0  tx(pkt,err): 0/4
'T1' 172.16.208.2:0  selectors(total,up): 1/0  rx(pkt,err): 0/0  tx(pkt,err): 0/0
```

```
# get router info routing-table static
Routing table for VRF=0
S       8.0.0.0/8 [1/0] via T5 tunnel 172.16.210.2, [1/0]
```

Traffic is using the backup overlay, T5:

```
# diagnose sniffer packet any 'host 8.8.8.8' 4
interfaces=[any]
filters=[host 8.8.8.8]
1.907944 port5 in 172.16.205.100 -> 8.8.8.8: icmp: echo request
1.908045 T5 out 172.16.205.100 -> 8.8.8.8: icmp: echo request
1.912283 T5 in 8.8.8.8 -> 172.16.205.100: icmp: echo reply
1.912351 port5 out 8.8.8.8 -> 172.16.205.100: icmp: echo reply
2.665921 port5 in 172.16.205.100 -> 8.8.8.8: icmp: echo request
2.665999 T5 out 172.16.205.100 -> 8.8.8.8: icmp: echo request
2.670209 T5 in 8.8.8.8 -> 172.16.205.100: icmp: echo reply
2.670235 port5 out 8.8.8.8 -> 172.16.205.100: icmp: echo reply
5.269997 port5 in 172.16.205.100 -> 8.8.8.8: icmp: echo request
5.270090 T5 out 172.16.205.100 -> 8.8.8.8: icmp: echo request
5.274275 T5 in 8.8.8.8 -> 172.16.205.100: icmp: echo reply
5.274300 port5 out 8.8.8.8 -> 172.16.205.100: icmp: echo reply
5.781848 port5 in 172.16.205.100 -> 8.8.8.8: icmp: echo request
5.781920 T5 out 172.16.205.100 -> 8.8.8.8: icmp: echo request
5.786334 T5 in 8.8.8.8 -> 172.16.205.100: icmp: echo reply
5.786363 port5 out 8.8.8.8 -> 172.16.205.100: icmp: echo reply
```

- If T4 recovers, T5 is deactivated and traffic switches to T4:

```
# get vpn ipsec tunnel summary
'T2' 172.16.203.2:0  selectors(total,up): 2/0  rx(pkt,err): 0/0  tx(pkt,err): 0/0
'T3' 172.16.206.2:0  selectors(total,up): 2/0  rx(pkt,err): 0/0  tx(pkt,err): 0/0
'T4' 172.16.209.2:0  selectors(total,up): 2/2  rx(pkt,err): 0/0  tx(pkt,err): 0/0
'T5' 172.16.210.2:0  selectors(total,up): 2/0  rx(pkt,err): 0/0  tx(pkt,err): 0/0
'T1' 172.16.208.2:0  selectors(total,up): 2/0  rx(pkt,err): 0/0  tx(pkt,err): 0/0
```

```
# get router info routing-table static
Routing table for VRF=0
S       8.0.0.0/8 [1/0] via T4 tunnel 172.16.209.2, [1/0]
```

The primary overlay T4 has recovered, and the backup overlay is down again:

```
# diagnose sniffer packet any 'host 8.8.8.8' 4
interfaces=[any]
filters=[host 8.8.8.8]
4.555685 port5 in 172.16.205.100 -> 8.8.8.8: icmp: echo request
4.555790 T4 out 172.16.205.100 -> 8.8.8.8: icmp: echo request
4.560428 T4 in 8.8.8.8 -> 172.16.205.100: icmp: echo reply
4.560478 port5 out 8.8.8.8 -> 172.16.205.100: icmp: echo reply
5.163223 port5 in 172.16.205.100 -> 8.8.8.8: icmp: echo request
5.163332 T4 out 172.16.205.100 -> 8.8.8.8: icmp: echo request
5.167590 T4 in 8.8.8.8 -> 172.16.205.100: icmp: echo reply
5.167620 port5 out 8.8.8.8 -> 172.16.205.100: icmp: echo reply
5.650089 port5 in 172.16.205.100 -> 8.8.8.8: icmp: echo request
5.650194 T4 out 172.16.205.100 -> 8.8.8.8: icmp: echo request
5.654352 T4 in 8.8.8.8 -> 172.16.205.100: icmp: echo reply
5.654387 port5 out 8.8.8.8 -> 172.16.205.100: icmp: echo reply
6.102181 port5 in 172.16.205.100 -> 8.8.8.8: icmp: echo request
6.102263 T4 out 172.16.205.100 -> 8.8.8.8: icmp: echo request
6.106411 T4 in 8.8.8.8 -> 172.16.205.100: icmp: echo reply
6.106445 port5 out 8.8.8.8 -> 172.16.205.100: icmp: echo reply
```

SD-WAN Cloud Assisted Monitoring service widgets - FAZ 7.4.1

New widgets are introduced in FortiAnalyzer 7.4.1 for the SD-WAN Cloud Assisted Monitoring service on FortiOS.

Topology

This feature requires an SD-WAN connected to the internet to run speed tests on SD-WAN member interfaces. The FortiGate version should be 7.4.1 or later, and SD-WAN Bandwidth Monitoring Result event logs are sent from FortiGate.

To run speed tests from the FortiGate devices:

Enter the following command to download the speed test server list from FortiGate Cloud:

```
exec speed-test-server download
```

Enter the following command to list all available servers:

```
exec speed-test-server list
```

Enter the following command to measure bandwidth on an interface to a test server:

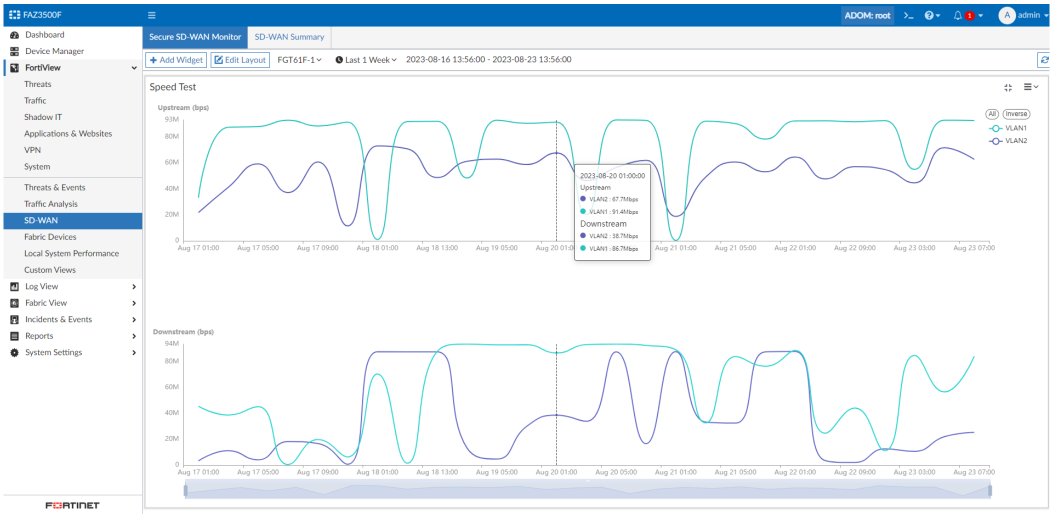
```
exec speed-test {auto | <outgoing interface name>} <server>
```

For more information, see the FortiGate / FortiOS Administration Guide.

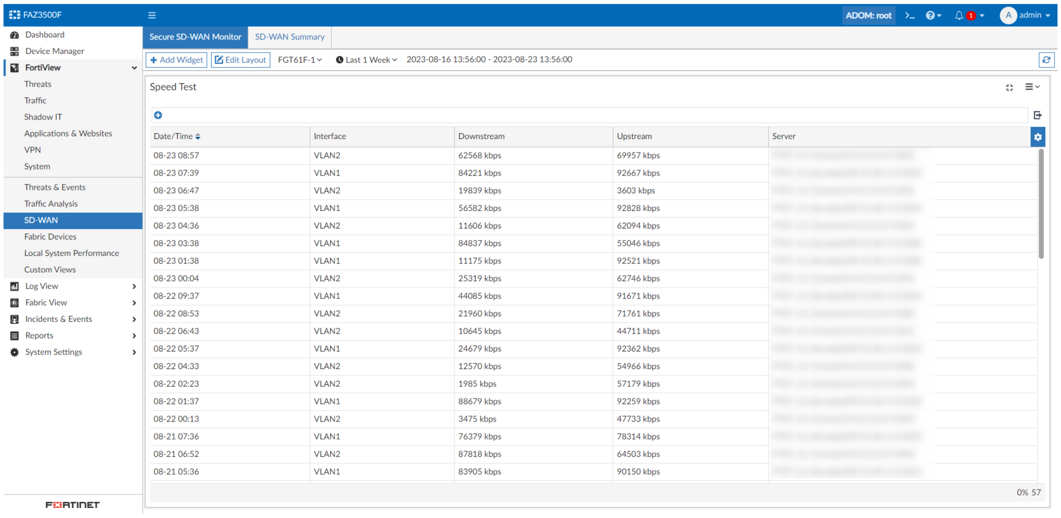
New SD-WAN Cloud Assisted Monitoring widgets and charts in FortiAnalyzer:

Speed Test is a new widget added to the *Secure SD-WAN Monitor* dashboard. This widget displays the download and upload speeds for all tests run on SD-WAN interfaces through the specified time period. You can select to display as a combined line chart or as a table chart.

The following is an example of the line chart for *Speed Test*:



The following is an example of the table for *Speed Test*:



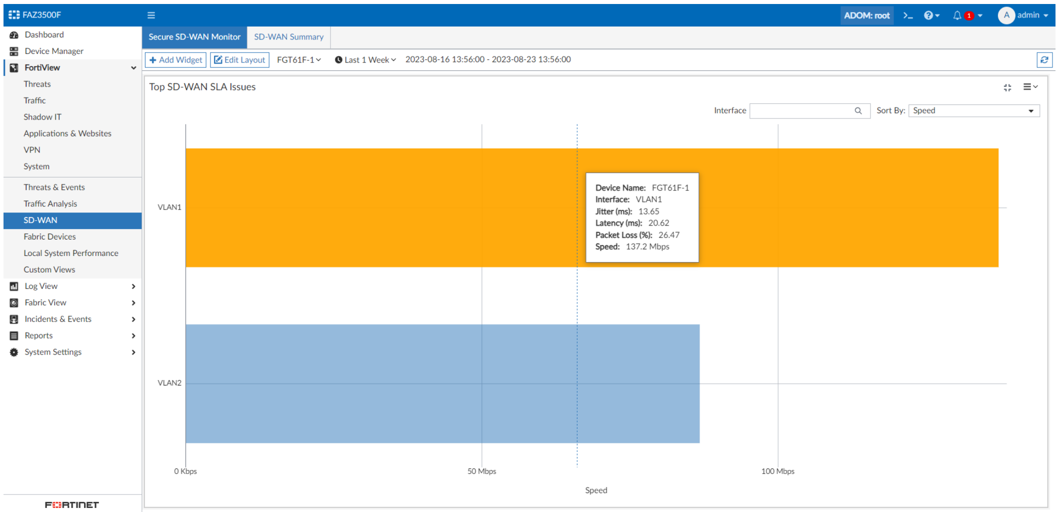
2023-08-16 13:56:00 - 2023-08-23 13:56:00

Speed Test

Date/Time	Interface	Downstream	Upstream	Server
08-23 08:57	VLAN2	62568 kbps	69957 kbps	[redacted]
08-23 07:39	VLAN1	84221 kbps	92667 kbps	[redacted]
08-23 06:47	VLAN2	19839 kbps	3603 kbps	[redacted]
08-23 05:38	VLAN1	56582 kbps	92828 kbps	[redacted]
08-23 04:36	VLAN2	11606 kbps	62094 kbps	[redacted]
08-23 03:38	VLAN1	84837 kbps	55046 kbps	[redacted]
08-23 01:38	VLAN1	11175 kbps	92521 kbps	[redacted]
08-23 00:04	VLAN2	25319 kbps	62746 kbps	[redacted]
08-22 09:37	VLAN1	44085 kbps	91671 kbps	[redacted]
08-22 08:53	VLAN2	21960 kbps	71761 kbps	[redacted]
08-22 06:43	VLAN2	10645 kbps	44711 kbps	[redacted]
08-22 05:37	VLAN1	24679 kbps	92362 kbps	[redacted]
08-22 04:33	VLAN2	12570 kbps	54966 kbps	[redacted]
08-22 02:23	VLAN2	1985 kbps	57179 kbps	[redacted]
08-22 01:37	VLAN1	88679 kbps	92259 kbps	[redacted]
08-22 00:13	VLAN2	3475 kbps	47733 kbps	[redacted]
08-21 07:36	VLAN1	76379 kbps	78314 kbps	[redacted]
08-21 06:52	VLAN2	87818 kbps	64503 kbps	[redacted]
08-21 05:36	VLAN1	83905 kbps	90150 kbps	[redacted]

0% 57

Sort By Speed is a new option added to the *Top SD-WAN SLA Issues* widget in the *Secure SD-WAN Monitor* dashboard. This option displays the peak speed run on SD-WAN interfaces through specified time period.



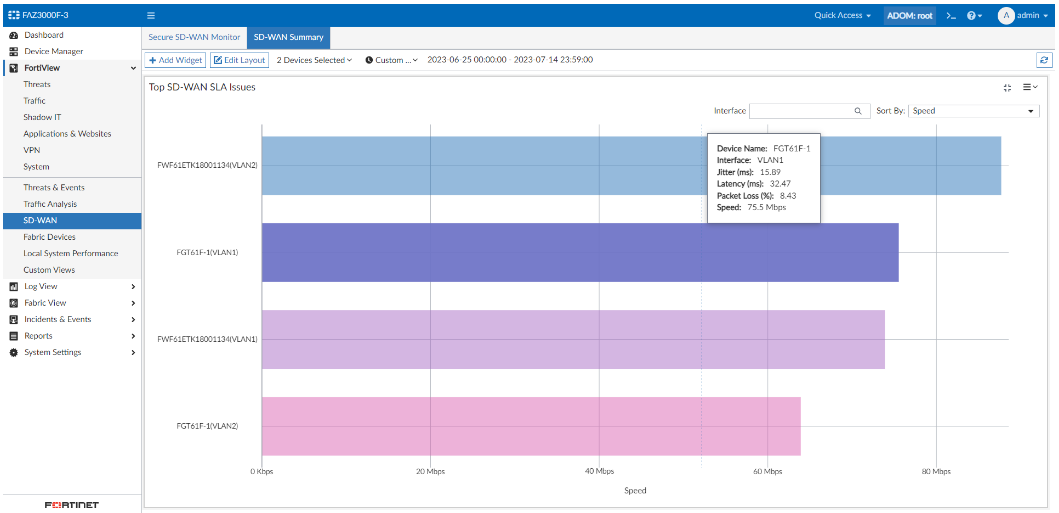
FGT61F-1 · Last 1 Week · 2023-08-16 13:56:00 - 2023-08-23 13:56:00

Top SD-WAN SLA Issues

Interface: · Sort By: Speed

Device Name: FGT61F-1
Interface: VLAN1
Jitter (ms): 13.65
Latency (ms): 20.62
Packet Loss (%): 26.47
Speed: 137.2 Mbps

The new *Sort By Speed* option is also added to the *Top SD-WAN SLA Issues* widget in the *SD-WAN Summary* dashboard. This option displays the peak speed run on SD-WAN interfaces through specified time period for selected devices.



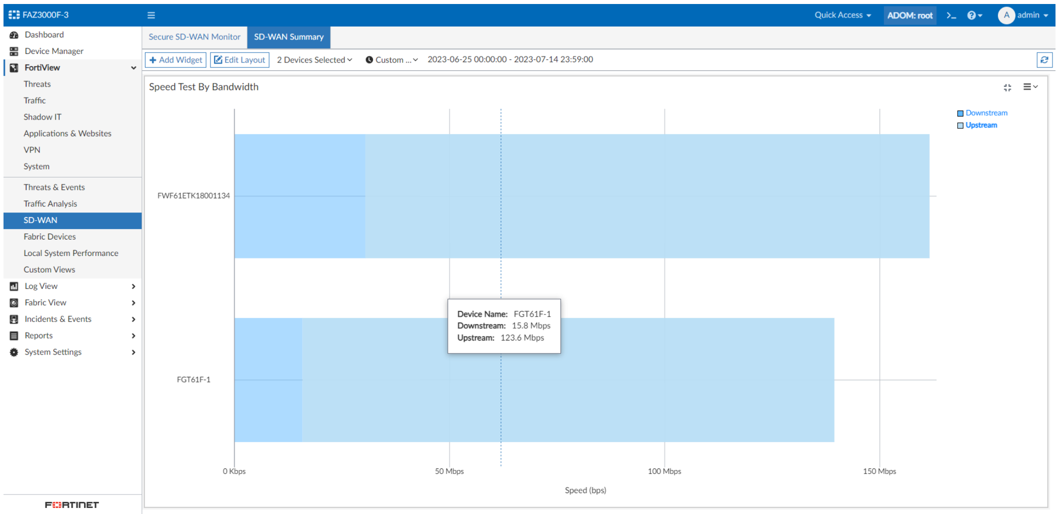
2 Devices Selected · Custom ... · 2023-06-25 00:00:00 - 2023-07-14 23:59:00

Top SD-WAN SLA Issues

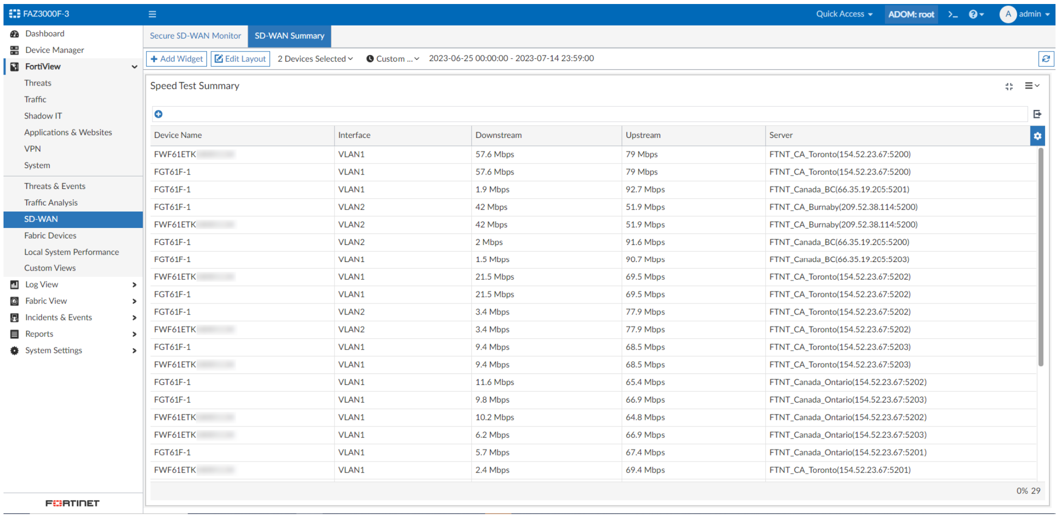
Interface: · Sort By: Speed

Device Name: FGT61F-1
Interface: VLAN1
Jitter (ms): 15.89
Latency (ms): 32.47
Packet Loss (%): 8.43
Speed: 75.5 Mbps

Speed Test By Bandwidth is a new widget added to the *SD-WAN Summary* dashboard. This widget displays a bar chart of the combined download and upload speeds for all SD-WAN interfaces on each device.



Speed Test Summary is a new widget added to the *SD-WAN Summary* dashboard. This widget displays a table of the download and upload speeds for all tests run on SD-WAN interfaces through specified time period on selected devices.

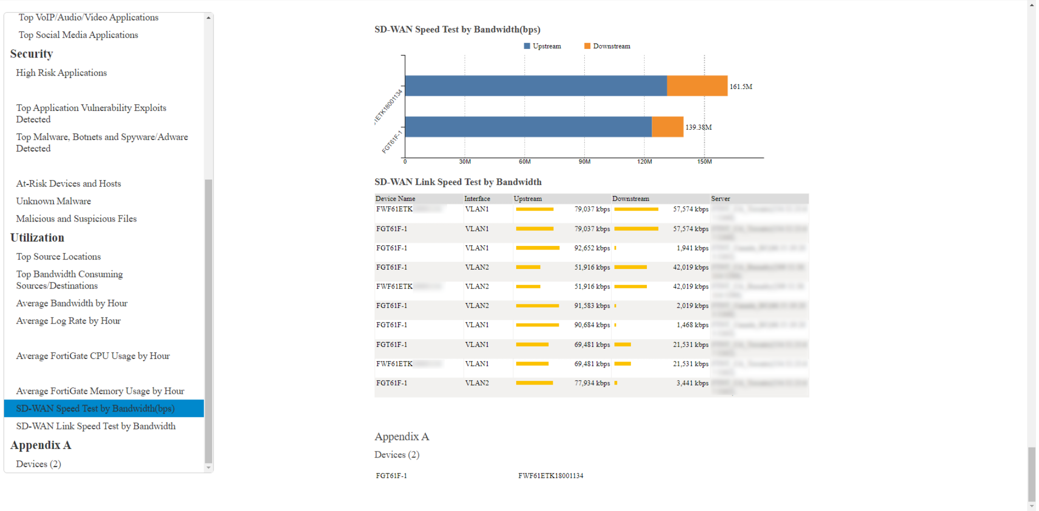


Device Name	Interface	Downstream	Upstream	Server
FWF61ETK[redacted]	VLAN1	57.6 Mbps	79 Mbps	FTNT_CA_Toronto(154.52.23.67:5200)
FGT61F-1	VLAN1	57.6 Mbps	79 Mbps	FTNT_CA_Toronto(154.52.23.67:5200)
FGT61F-1	VLAN1	1.9 Mbps	92.7 Mbps	FTNT_Canada_BC(66.35.19.205:5201)
FGT61F-1	VLAN2	42 Mbps	51.9 Mbps	FTNT_CA_Burnaby(209.52.38.114:5200)
FWF61ETK[redacted]	VLAN2	42 Mbps	51.9 Mbps	FTNT_CA_Burnaby(209.52.38.114:5200)
FGT61F-1	VLAN2	2 Mbps	91.6 Mbps	FTNT_Canada_BC(66.35.19.205:5200)
FGT61F-1	VLAN1	1.5 Mbps	90.7 Mbps	FTNT_Canada_BC(66.35.19.205:5203)
FWF61ETK[redacted]	VLAN1	21.5 Mbps	69.5 Mbps	FTNT_CA_Toronto(154.52.23.67:5202)
FGT61F-1	VLAN1	21.5 Mbps	69.5 Mbps	FTNT_CA_Toronto(154.52.23.67:5202)
FGT61F-1	VLAN2	3.4 Mbps	77.9 Mbps	FTNT_CA_Toronto(154.52.23.67:5202)
FWF61ETK[redacted]	VLAN2	3.4 Mbps	77.9 Mbps	FTNT_CA_Toronto(154.52.23.67:5202)
FGT61F-1	VLAN1	9.4 Mbps	68.5 Mbps	FTNT_CA_Toronto(154.52.23.67:5203)
FWF61ETK[redacted]	VLAN1	9.4 Mbps	68.5 Mbps	FTNT_CA_Toronto(154.52.23.67:5203)
FGT61F-1	VLAN1	11.6 Mbps	65.4 Mbps	FTNT_Canada_Ontario(154.52.23.67:5202)
FGT61F-1	VLAN1	9.8 Mbps	66.9 Mbps	FTNT_Canada_Ontario(154.52.23.67:5203)
FWF61ETK[redacted]	VLAN1	10.2 Mbps	64.8 Mbps	FTNT_Canada_Ontario(154.52.23.67:5202)
FWF61ETK[redacted]	VLAN1	6.2 Mbps	66.9 Mbps	FTNT_Canada_Ontario(154.52.23.67:5203)
FGT61F-1	VLAN1	5.7 Mbps	67.4 Mbps	FTNT_Canada_Ontario(154.52.23.67:5201)
FWF61ETK[redacted]	VLAN1	2.4 Mbps	69.4 Mbps	FTNT_CA_Toronto(154.52.23.67:5201)

0% 29

An *SD-WAN Speed Test By Bandwidth(bps)* bar chart is added to the *Secure SD-WAN Assessment Report*. This chart displays the combined download and upload speeds for all SD-WAN interfaces on each device.

A *SD-WAN Link Speed Test by Bandwidth* table is also added to the *Secure SD-WAN Assessment Report*. This table displays the download and upload speeds for all tests run on SD-WAN interfaces through the specified time period on selected devices



Top VoIP/Audio/Video Applications
Top Social Media Applications
Security
High Risk Applications
Top Application Vulnerability Exploits Detected
Top Malware, Botnets and Spyware/Adware Detected
At-Risk Devices and Hosts
Unknown Malware
Malicious and Suspicious Files
Utilization
Top Source Locations
Top Bandwidth Consuming Sources/Destinations
Average Bandwidth by Hour
Average Log Rate by Hour
Average FortiGate CPU Usage by Hour
Average FortiGate Memory Usage by Hour
SD-WAN Speed Test by Bandwidth(bps)
SD-WAN Link Speed Test by Bandwidth
Appendix A
Devices (2)

SD-WAN Speed Test by Bandwidth(bps)

SD-WAN Link Speed Test by Bandwidth

Device Name	Interface	Upstream	Downstream	Server
FWF61ETK[redacted]	VLAN1	79,037 kbps	57,574 kbps	[redacted]
FGT61F-1	VLAN1	79,037 kbps	57,574 kbps	[redacted]
FGT61F-1	VLAN1	92,652 kbps	1,941 kbps	[redacted]
FGT61F-1	VLAN2	51,916 kbps	42,019 kbps	[redacted]
FWF61ETK[redacted]	VLAN2	51,916 kbps	42,019 kbps	[redacted]
FGT61F-1	VLAN2	91,583 kbps	2,019 kbps	[redacted]
FGT61F-1	VLAN1	90,684 kbps	1,468 kbps	[redacted]
FGT61F-1	VLAN1	69,481 kbps	21,531 kbps	[redacted]
FWF61ETK[redacted]	VLAN1	69,481 kbps	21,531 kbps	[redacted]
FGT61F-1	VLAN2	77,934 kbps	3,441 kbps	[redacted]

Appendix A

Devices (2)

FGT61F-1	FWF61ETK18001134

SD-WAN Monitoring dashboards allow full widget customization - FMG 7.4.2

This information is also available in the FortiManager 7.4 Administration Guide:

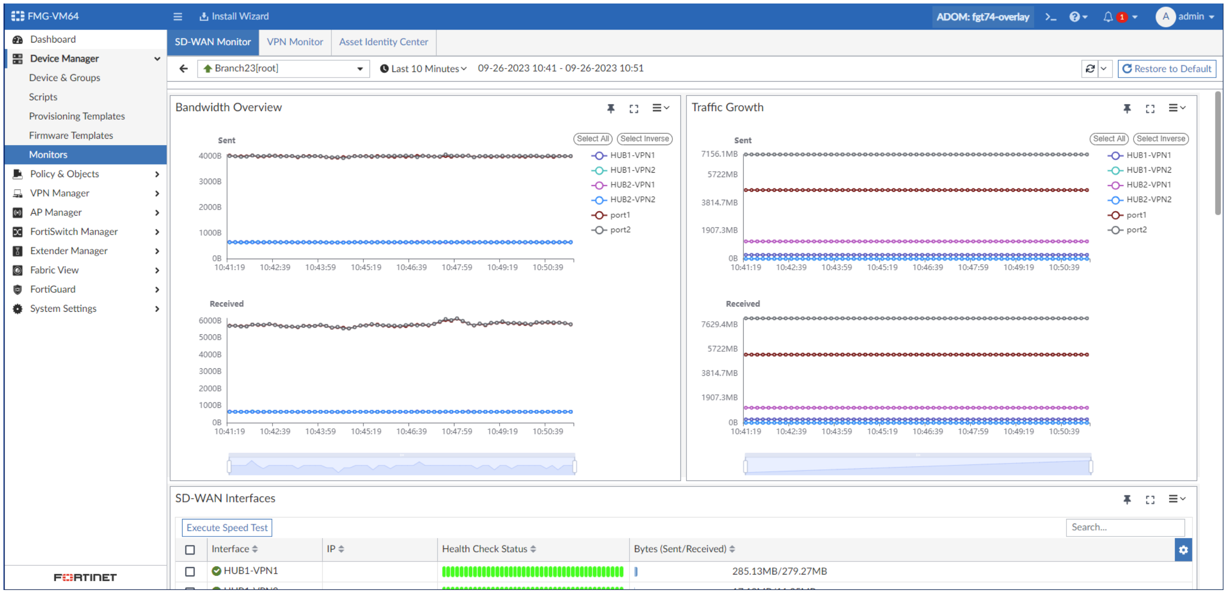
- SD-WAN Monitor Table View

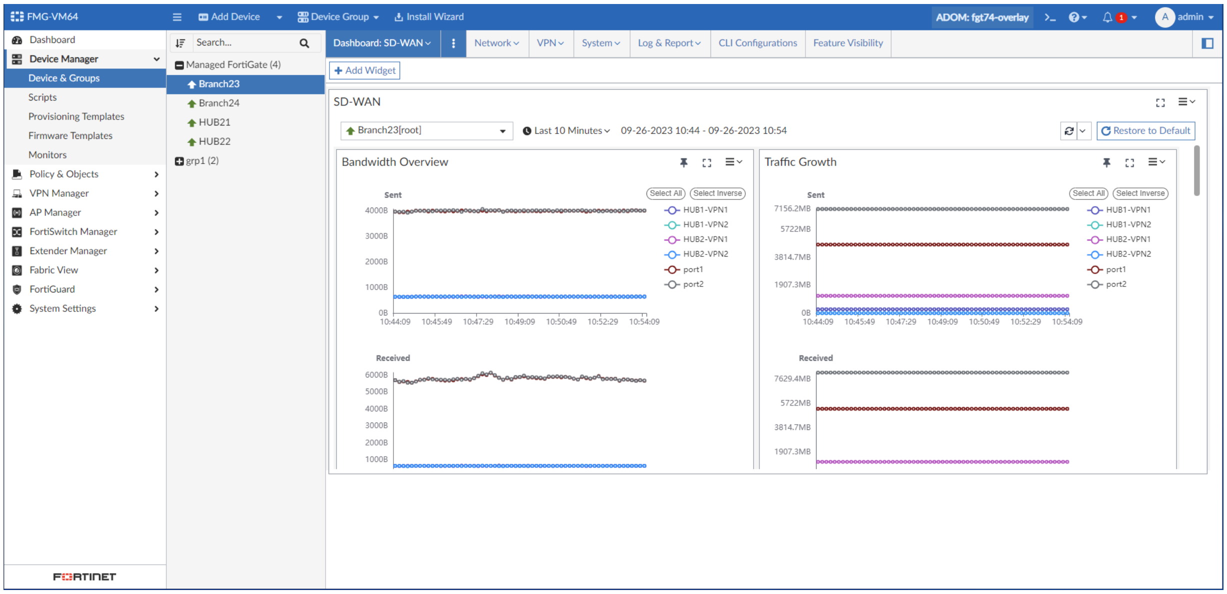
SD-WAN Monitoring dashboards allow full widget customization (movable and full-screen options).

To view customization options for SD-WAN monitoring dashboards:

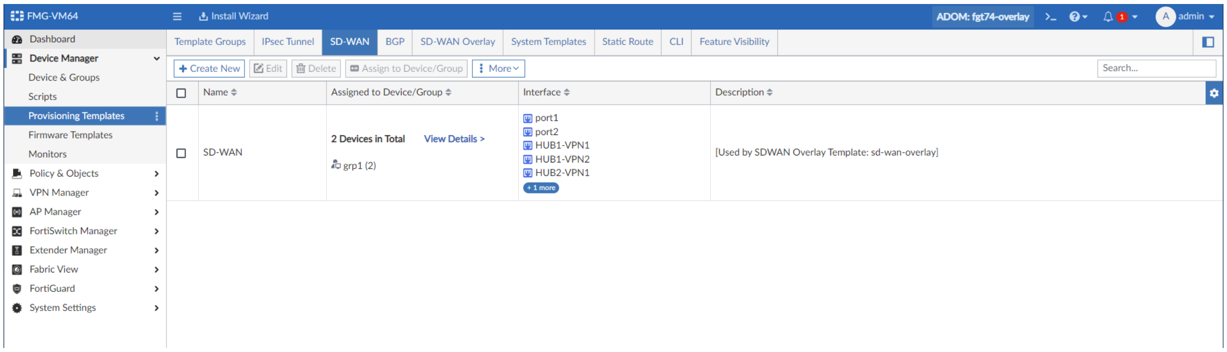
1. Go to *Device Manager* > *Monitors* > *SD-WAN Monitor* or *Device Manager Device & Groups* > *[Managed Device]* > *Dashboard* > *SD-WAN*.

- History view supports a dashboard layout, and all widgets can be dragged and repositioned.
- Features such as refreshing widgets and switching devices are supported.

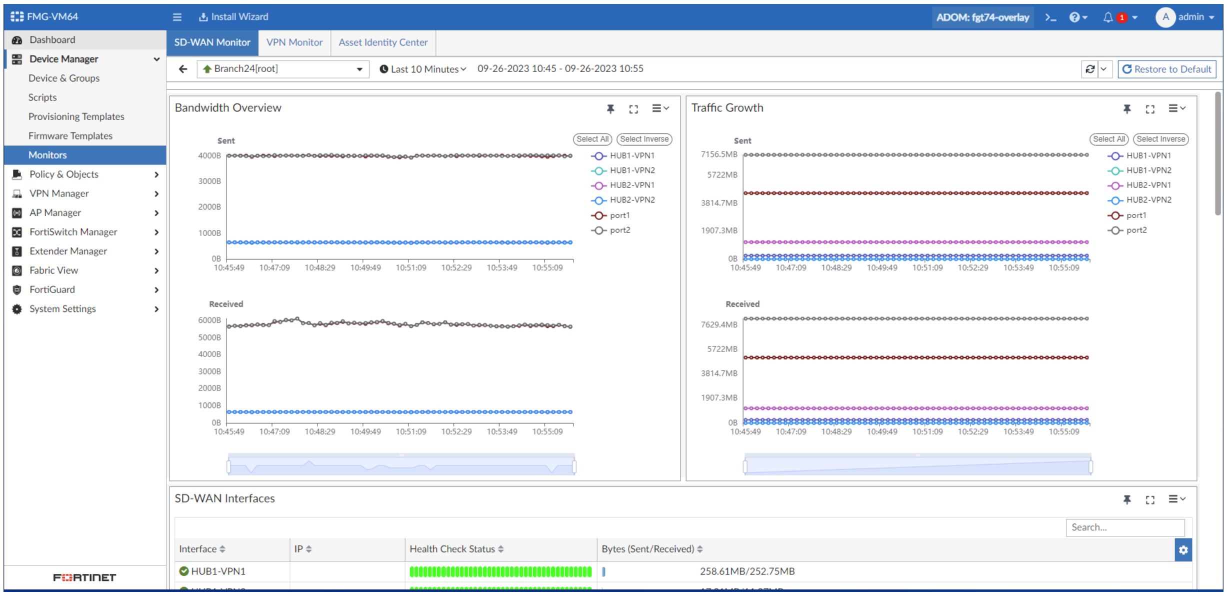




2. The widget layout persists based on either the device configuration or template configuration.
   - All SD-WAN devices assigned to the same SD-WAN template use the same layout.
   - If an SD-WAN device is not part of an SD-WAN template, then it uses its own layout.
   - Currently, this only applies to devices/groups that are assigned to the SD-WAN template directly, and not through Template Groups or the SD-WAN Overlay Template.



□	Name	Assigned to Device/Group	Interface	Description
□	SD-WAN	2 Devices in Total View Details > grp1 (2)	port1 port2 HUB1-VPN1 HUB1-VPN2 HUB2-VPN1 + 1 more	[Used by SDWAN Overlay Template: sd-wan-overlay]



Interface	IP	Health Check Status	Bytes (Sent/Received)
HUB1-VPN1			258.61MB/252.75MB

Provisioning

7.4.0

- Automated SD-WAN overlay process adds "branch_id" meta variable auto assignment FMG on page 83
- SD-WAN Monitoring Map integrates with Cloud Assisted Monitoring Service to allow FGT interface speed tests from inside FMG FMG on page 48
- Fortinet factory-default wireless and extender templates FMG on page 85
- SD-WAN template for heterogeneous WAN link types FMG on page 92

7.4.1

- Support the new SD-WAN Overlay-as-a-Service 7.4.1 on page 94
- Fabric Authorization Template is integrated with Device Blueprint and supports meta variables FMG 7.4.1 on page 97

7.4.2

- Export Managed FortiAPs and import FortiAPs from a CSV file FMG 7.4.2 on page 100
- Meta variables are available in the SSID, FortiSwitch VLANs, and FortiSwitch Templates configuration FMG 7.4.2 on page 102
- FortiSwitch devices can be imported from a CSV file FMG 7.4.2 on page 107
- Factory default SSIDs and AP Profiles configuration updated FMG 7.4.2 on page 109

Automated SD-WAN post overlay process creates policies to allow the health-check traffic to flow between Branch and HUB - FMG

This information is also available in the FortiManager 7.4 Administration Guide:

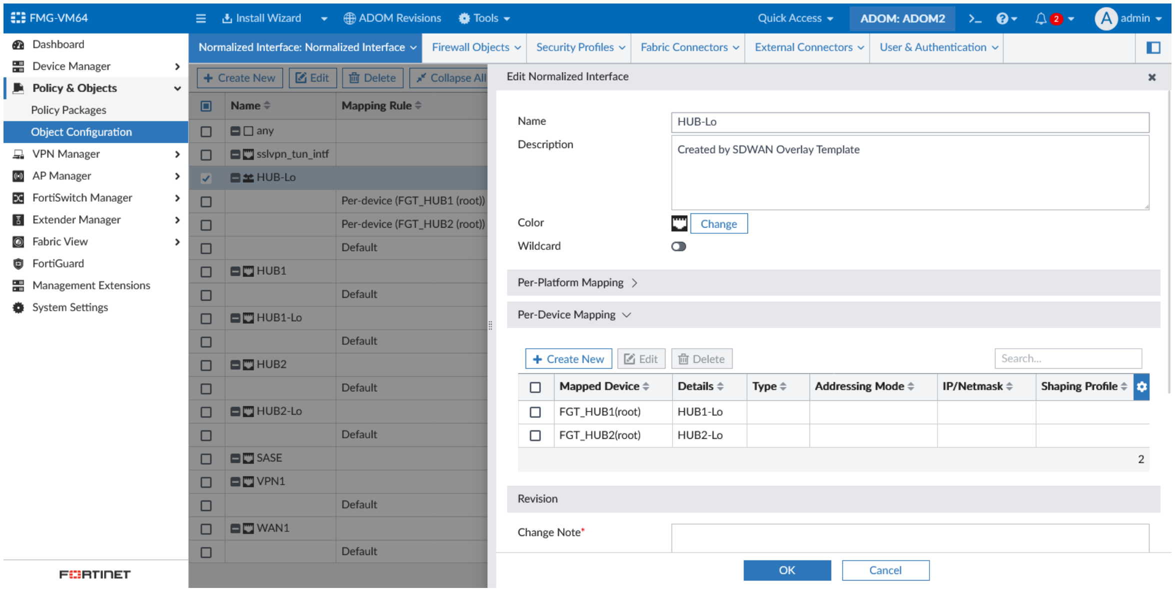
- Configuring an SD-WAN overlay template

Automated SD-WAN post overlay process creates policies to allow the health-checks traffic to flow between Branch and HUB.

The SD-WAN overlay template includes two new options in the wizard to automate the post-wizard processes. The SD-WAN overlay template example configured in this document uses a dual-hub topology.

1. **Normalize Interfaces**
   Enable the *Normalize Interfaces* option to normalize the SD-WAN zones created by the template.

- The following normalized interface is created for the SD-WAN Hub(s):
  - *HUB-Lo* with *Per-Device Mapping* to *HUB1-Lo* for the HUB 1 device and *HUB2-Lo* from the HUB 2 device.



Name	Mapping Rule
any	
sslvpn_tun_intf	
HUB-Lo	
	Per-device (FGT_HUB1 (root))
	Per-device (FGT_HUB2 (root))
	Default
HUB1	
	Default
HUB1-Lo	
	Default
HUB2	
	Default
HUB2-Lo	
	Default
SASE	
VPN1	
	Default
WAN1	
	Default

Edit Normalized Interface

Name	HUB-Lo
Description	Created by SDWAN Overlay Template
Color	Change
Wildcard	

Per-Platform Mapping

Per-Device Mapping

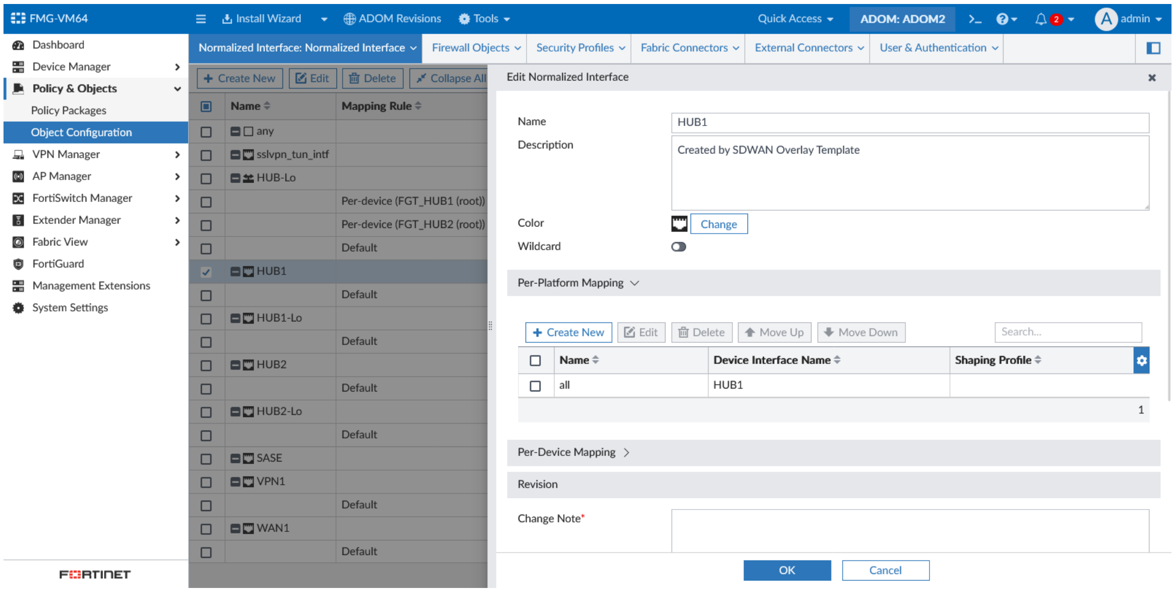
Mapped Device	Details	Type	Addressing Mode	IP/Netmask	Shaping Profile
FGT_HUB1(root)	HUB1-Lo				
FGT_HUB2(root)	HUB2-Lo				

2

Revision

Change Note*

- The following normalized interfaces are created for branch devices:
  - The *HUB1 SD-WAN zone* is mapped per-platform to *HUB1*.



Name	Mapping Rule
any	
sslvpn_tun_intf	
HUB-Lo	
	Per-device (FGT_HUB1 (root))
	Per-device (FGT_HUB2 (root))
	Default
HUB1	
	Default
HUB1-Lo	
	Default
HUB2	
	Default
HUB2-Lo	
	Default
SASE	
VPN1	
	Default
WAN1	
	Default

Edit Normalized Interface

Name	HUB1
Description	Created by SDWAN Overlay Template
Color	Change
Wildcard	

Per-Platform Mapping

Name	Device Interface Name	Shaping Profile
all	HUB1	

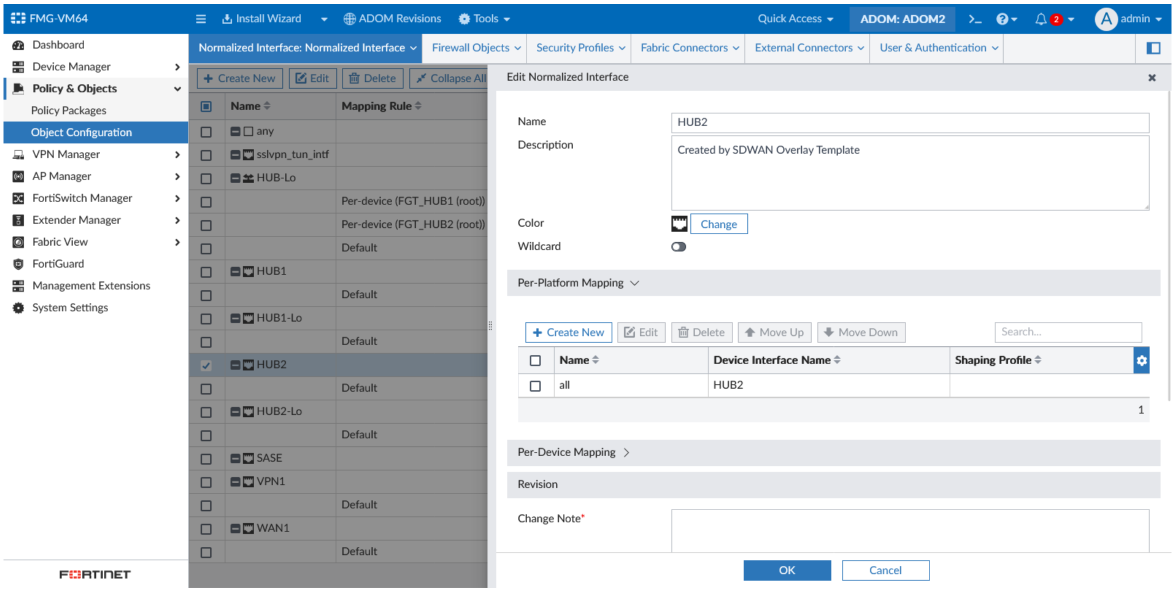
1

Per-Device Mapping

Revision

Change Note*

- The *HUB2 SD-WAN zone* is mapped per-platform to *HUB2*.



Name	Mapping Rule
any	
sslvpn_tun_intf	
HUB-Lo	
	Per-device (FGT_HUB1 (root))
	Per-device (FGT_HUB2 (root))
	Default
HUB1	
	Default
HUB1-Lo	
	Default
☑ HUB2	
	Default
HUB2-Lo	
	Default
SASE	
VPN1	
	Default
WAN1	
	Default

Edit Normalized Interface

Name: HUB2

Description: Created by SDWAN Overlay Template

Color: Change

Wildcard

Per-Platform Mapping

Create New | Edit | Delete | Move Up | Move Down | Search...

Name	Device Interface Name	Shaping Profile
all	HUB2	

1

Per-Device Mapping

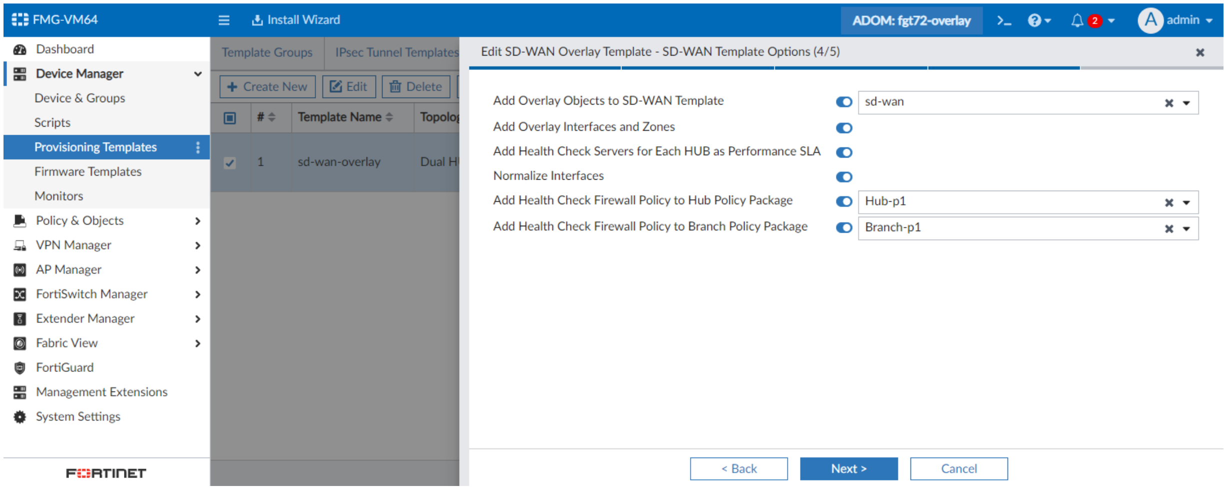
Revision

Change Note*

OK | Cancel

- VPN IPsec tunnel templates are created for HUB interfaces when using the SD-WAN overlay template. When *Normalized Interface* is enabled, normalized interfaces for the VPNs are added to the normalized interface list.

2. **Add Health Check Firewall Policy to Hub/Branch Policy Package**
   Enable the *Add Health Check Firewall Policy to Hub/Branch Policy Package* option to create health check firewall policies (or policy blocks) for HUB(s) and branches.
   - Users must select the HUB and branch policy package that will be used during the wizard configuration. You can select an existing policy package or create a new one.



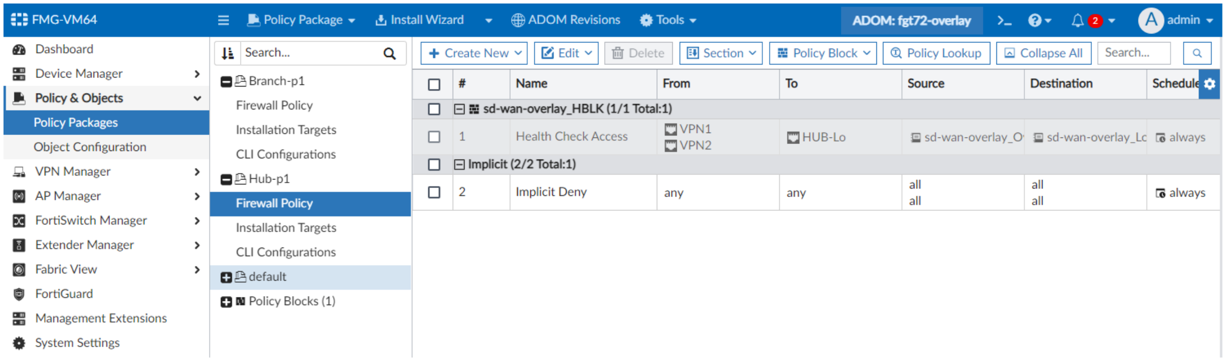
Edit SD-WAN Overlay Template - SD-WAN Template Options (4/5)

Add Overlay Objects to SD-WAN Template	●	sd-wan
Add Overlay Interfaces and Zones	●	
Add Health Check Servers for Each HUB as Performance SLA	●	
Normalize Interfaces	●	
Add Health Check Firewall Policy to Hub Policy Package	●	Hub-p1
Add Health Check Firewall Policy to Branch Policy Package	●	Branch-p1

< Back | Next > | Cancel

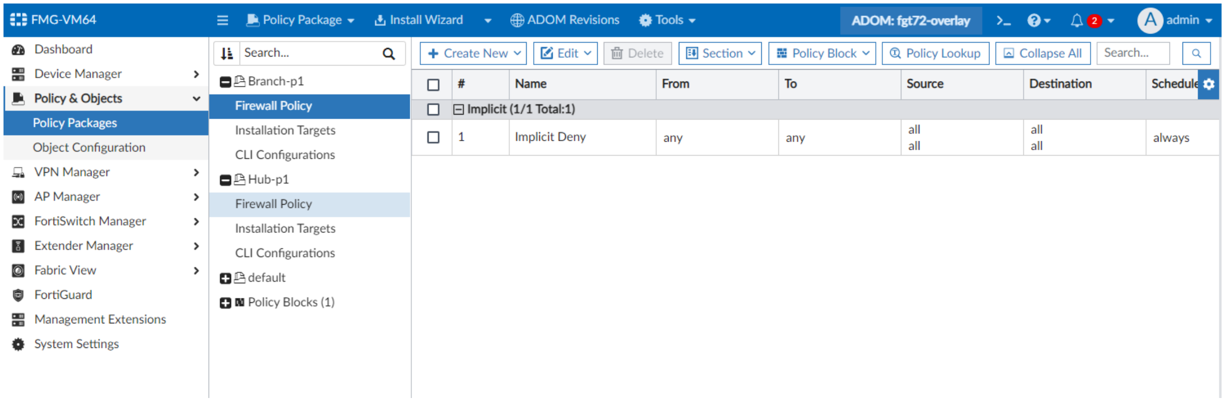
   - Based on the selection, firewall policies (or policy blocks) are created to allow SLA health checks to each device loopback.

- The SD-WAN overlay template creates the policy block and applies it to the top of the HUB Policy Package.



□	#	Name	From	To	Source	Destination	Schedule
□	sd-wan-overlay_HBLK (1/1 Total:1)						
□	1	Health Check Access	VPN1 VPN2	HUB-Lo	sd-wan-overlay_O	sd-wan-overlay_Lc	always
□	Implicit (2/2 Total:1)						
□	2	Implicit Deny	any	any	all all	all all	always

- A policy block is not created for the SD-WAN branch Policy Package.



□	#	Name	From	To	Source	Destination	Schedule
□	Implicit (1/1 Total:1)						
□	1	Implicit Deny	any	any	all all	all all	always

Automated SD-WAN overlay process adds "branch_id" meta variable auto assignment - FMG

This information is also available in the FortiManager 7.4 Administration Guide:

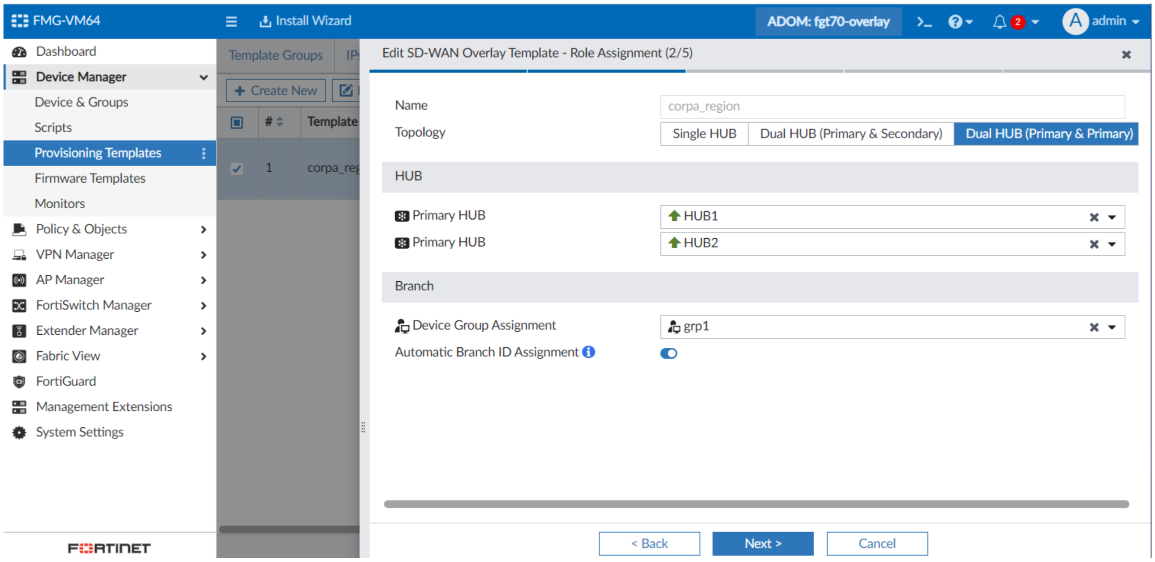
- Configuring an SD-WAN overlay template
- Objects and templates created by the SD-WAN overlay template

The automated SD-WAN overlay process adds "branch_id" meta variable auto assignment.

To automatically assign branch IDs using the SD-WAN overlay template:

1. Go to *Device Manager* > *Provisioning Templates* > *SD-WAN Overlay Template*.
2. Create or edit an SD-WAN overlay template.

3. On the *Role Assignment (2/5)* step in the wizard, enable *Automatic Branch ID Assignment*.



Edit SD-WAN Overlay Template - Role Assignment (2/5)

Name	corpa_region
Topology	Single HUB / Dual HUB (Primary & Secondary) / [✓ Dual HUB (Primary & Primary)]

HUB

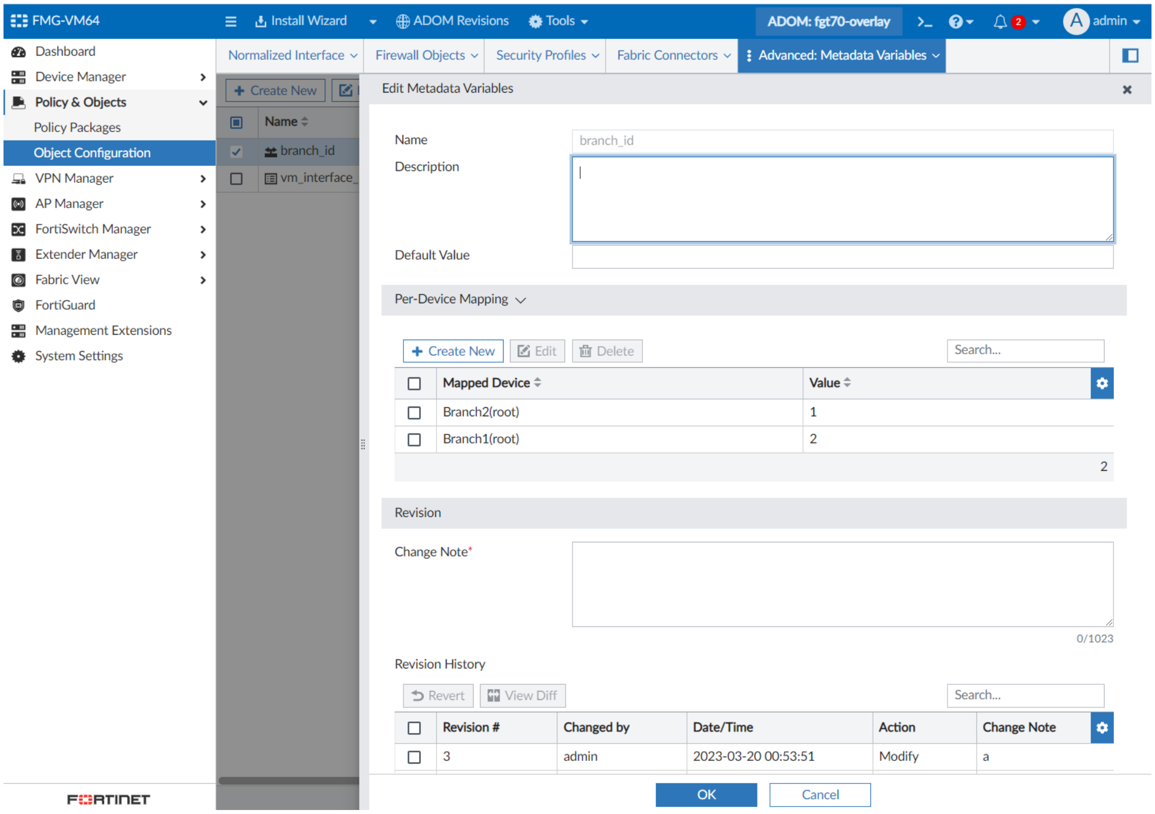
Primary HUB	HUB1
Primary HUB	HUB2

Branch

Device Group Assignment	grp1
Automatic Branch ID Assignment	●

< Back | Next > | Cancel

- When *Automatic Branch ID Assignment* is enabled, FortiManager automatically assigns and tracks a branch ID for each device in the branch device group. This also applies to devices added to the branch device group in the future, as well as those added to the device group using a zero-touch provisioning device blueprint.



Edit Metadata Variables

Name	branch_id
Description	
Default Value	

Per-Device Mapping

Create New | Edit | Delete | Search...

Mapped Device	Value
Branch2(root)	1
Branch1(root)	2

2

Revision

Change Note*

0/1023

Revision History

Revert | View Diff | Search...

Revision #	Changed by	Date/Time	Action	Change Note
3	admin	2023-03-20 00:53:51	Modify	a

OK | Cancel

- Branch ID values are between one and the maximum number allowed by the subnet. For example, the default 10.10.0.0/255.255.0.0 overlay network uses the /19 subnet when your setup includes 5 - 8 overlays. The maximum allowed branch IDs in this range is 8,190 based on the maximum number of number of usable IPs/FortiGates supported per overlay.

Fortinet factory-default wireless and extender templates - FMG

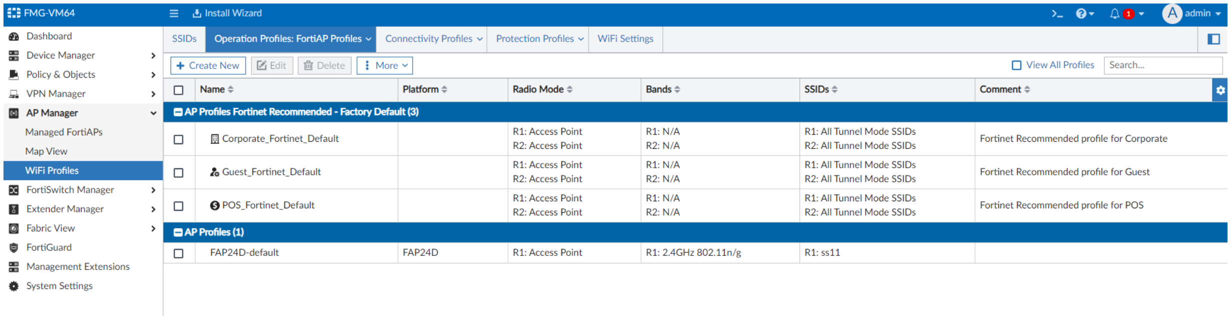
FortiManager includes Fortinet factory-default wireless and extender templates with built-in security and network configuration based on best security practices.

This information is also available in the FortiManager 7.4 Administration Guide:

- Using Fortinet recommended FortiAP and SSID profiles
- Using Fortinet recommended extender profiles

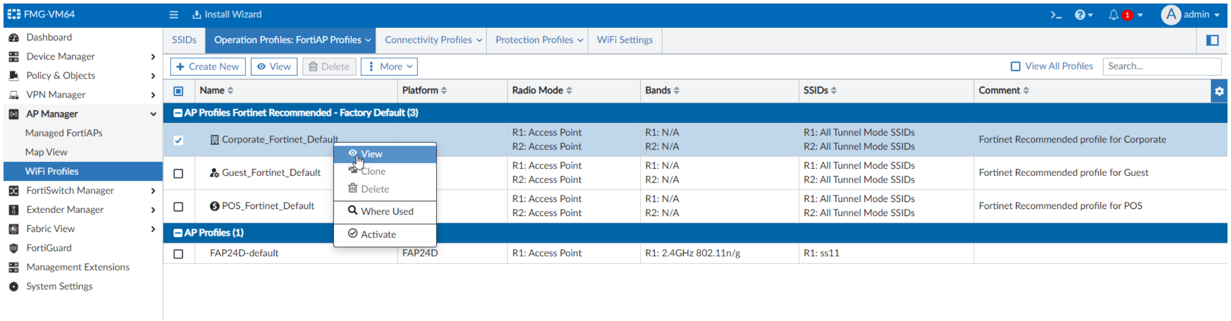
To use default FortiAP profiles and SSIDs:

1. Recommended FortiAP profiles are available in the FortiManager *AP Manager*.
   a. Go to *AP Manager > WiFi Profiles* under the *Operation Profiles > FortiAP Profiles* tab.

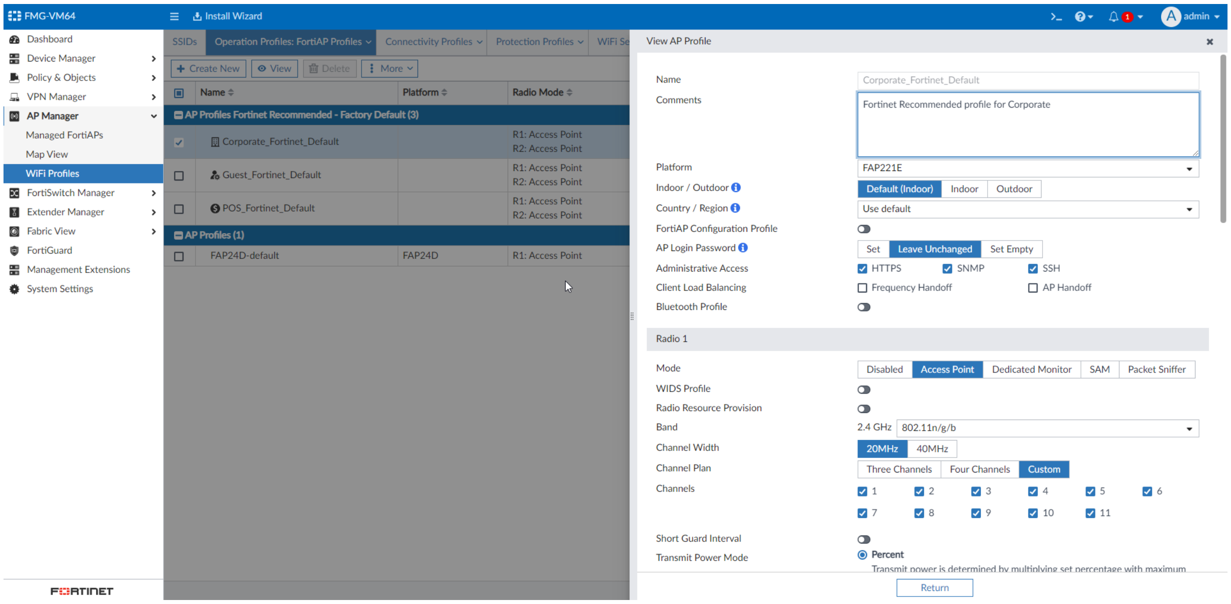


□	Name	Platform	Radio Mode	Bands	SSIDs	Comment
AP Profiles Fortinet Recommended - Factory Default (3)						
□	Corporate_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for Corporate
□	Guest_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for Guest
□	POS_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for POS
AP Profiles (1)						
□	FAP24D-default	FAP24D	R1: Access Point	R1: 2.4GHz 802.11n/g	R1: ss11	

b. Right click on a recommended AP profile and click *View*.



Name	Platform	Radio Mode	Bands	SSIDs	Comment
AP Profiles Fortinet Recommended - Factory Default (3)					
☑ Corporate_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for Corporate
□ Guest_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for Guest
□ POS_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for POS
AP Profiles (1)					
□ FAP24D-default	FAP24D	R1: Access Point	R1: 2.4GHz 802.11n/g	R1: ss11	



View AP Profile

Name	Corporate_Fortinet_Default
Comments	Fortinet Recommended profile for Corporate
Platform	FAP221E
Indoor / Outdoor	[✓ Default (Indoor)] Indoor Outdoor
Country / Region	Use default
FortiAP Configuration Profile	○
AP Login Password	Set [✓ Leave Unchanged] Set Empty
Administrative Access	☑ HTTPS ☑ SNMP ☑ SSH
Client Load Balancing	□ Frequency Handoff □ AP Handoff
Bluetooth Profile	○

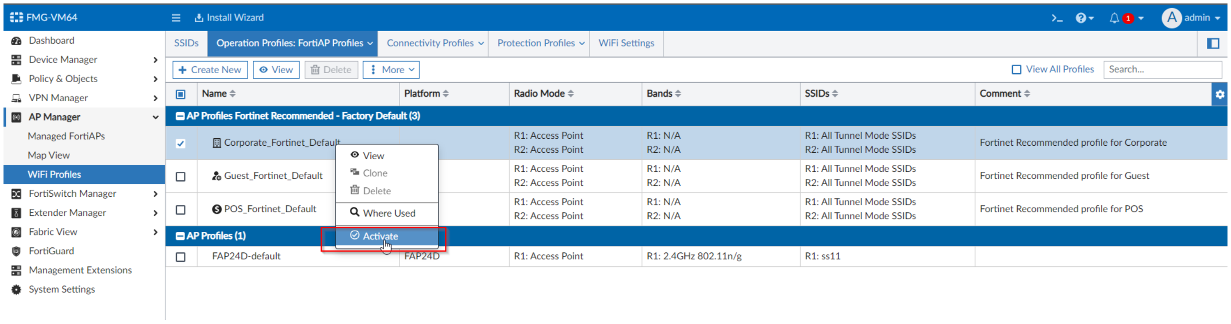
Radio 1

Mode	Disabled [✓ Access Point] Dedicated Monitor SAM Packet Sniffer
WIDS Profile	○
Radio Resource Provision	○
Band	2.4 GHz 802.11n/g/b
Channel Width	[✓ 20MHz] 40MHz
Channel Plan	Three Channels Four Channels [✓ Custom]
Channels	☑ 1 ☑ 2 ☑ 3 ☑ 4 ☑ 5 ☑ 6 ☑ 7 ☑ 8 ☑ 9 ☑ 10 ☑ 11
Short Guard Interval	○
Transmit Power Mode	● Percent

Transmit power is determined by multiplying set percentage with maximum

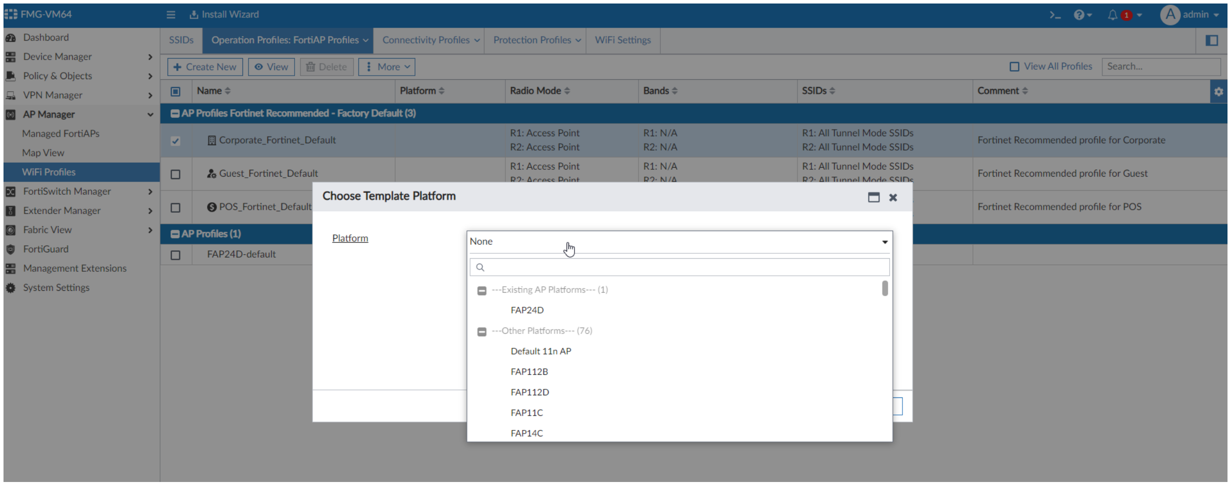
2. FortiAP profiles based on the recommended profiles can be created by activating the recommended profiles.

a. Right-click on a recommended profile and click *Activate*.



Name	Platform	Radio Mode	Bands	SSIDs	Comment
AP Profiles Fortinet Recommended - Factory Default (3)					
☑ Corporate_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for Corporate
□ Guest_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for Guest
□ POS_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for POS
AP Profiles (1)					
□ FAP24D-default	FAP24D	R1: Access Point	R1: 2.4GHz 802.11n/g	R1: ss11	

b. Select the platform type for the profile.



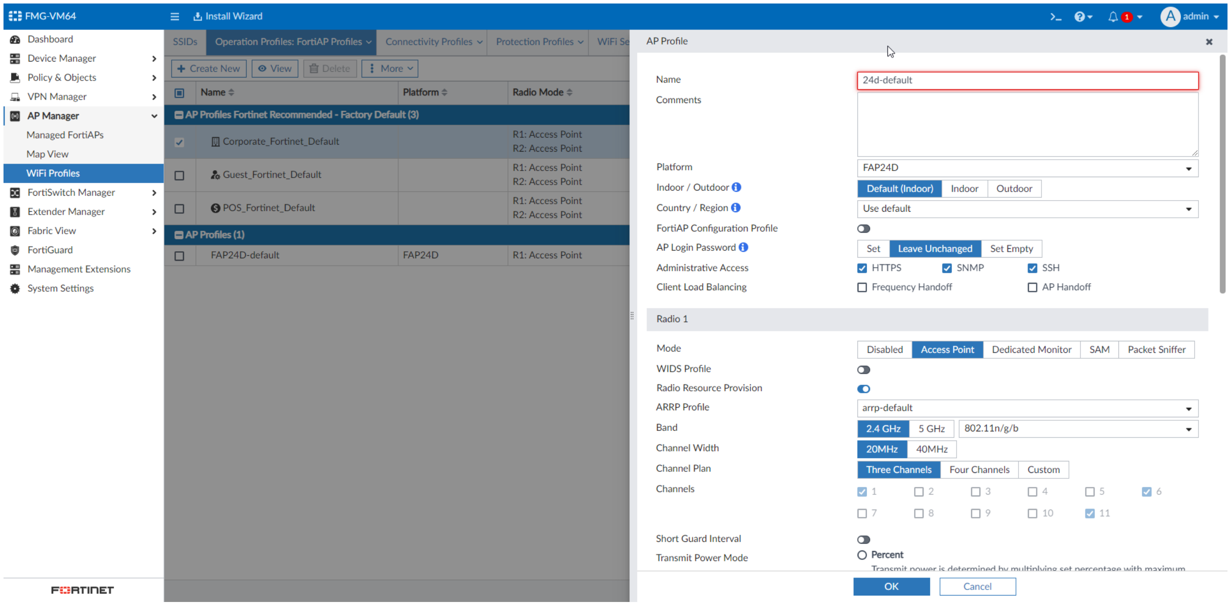
	Name	Platform	Radio Mode	Bands	SSIDs	Comment
AP Profiles Fortinet Recommended - Factory Default (3)						
☑	Corporate_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for Corporate
□	Guest_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for Guest
□	POS_Fortinet_Default					Fortinet Recommended profile for POS
AP Profiles (1)						
□	FAP24D-default					

Choose Template Platform

Platform: None

---Existing AP Platforms--- (1)
FAP24D
---Other Platforms--- (76)
Default 11n AP
FAP112B
FAP112D
FAP11C
FAP14C

c. Enter a name for the AP profile and configure the remaining settings if required.



	Name	Platform	Radio Mode
AP Profiles Fortinet Recommended - Factory Default (3)			
☑	Corporate_Fortinet_Default		R1: Access Point R2: Access Point
□	Guest_Fortinet_Default		R1: Access Point R2: Access Point
□	POS_Fortinet_Default		R1: Access Point R2: Access Point
AP Profiles (1)			
□	FAP24D-default	FAP24D	R1: Access Point

AP Profile

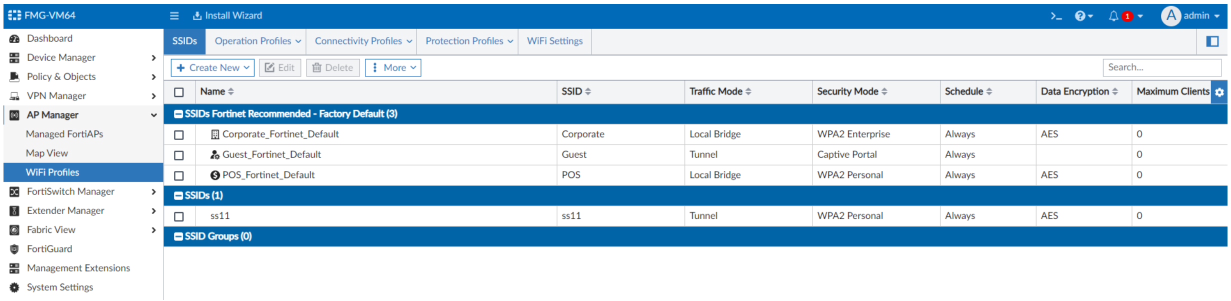
Name: 24d-default
Comments:
Platform: FAP24D
Indoor / Outdoor: [✓ Default (Indoor)] Indoor Outdoor
Country / Region: Use default
FortiAP Configuration Profile: ○
AP Login Password: Set [✓ Leave Unchanged] Set Empty
Administrative Access: ☑ HTTPS ☑ SNMP ☑ SSH
Client Load Balancing: □ Frequency Handoff □ AP Handoff

Radio 1

Mode: Disabled [✓ Access Point] Dedicated Monitor SAM Packet Sniffer
WIDS Profile: ○
Radio Resource Provision: ●
ARRP Profile: arrp-default
Band: [✓ 2.4 GHz] 5 GHz 802.11n/g/b
Channel Width: [✓ 20MHz] 40MHz
Channel Plan: [✓ Three Channels] Four Channels Custom
Channels: ☑ 1 □ 2 □ 3 □ 4 □ 5 ☑ 6 □ 7 □ 8 □ 9 □ 10 ☑ 11
Short Guard Interval: ○
Transmit Power Mode: ○ Percent
Transmit power is determined by multiplying set percentage with maximum

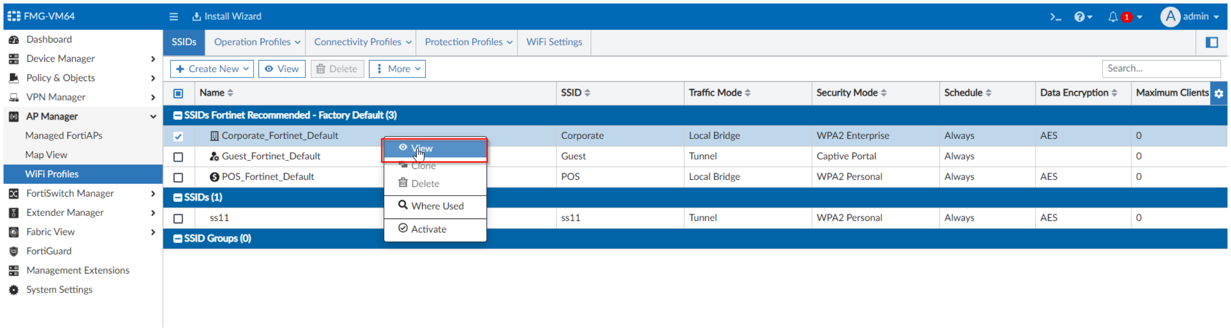
3. The recommended default AP SSIDs are shown in *AP Manager* in the *SSIDs* tab.

a. Go to *WiFi Profiles* and click the *SSIDs* tab to view the default SSIDs.

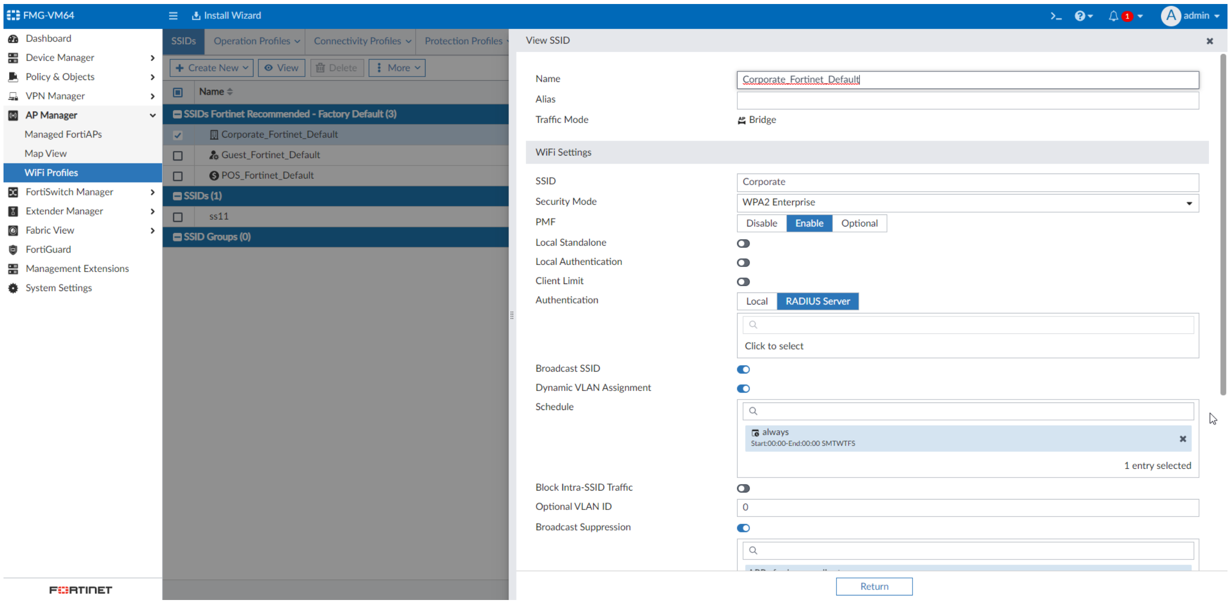


	Name	SSID	Traffic Mode	Security Mode	Schedule	Data Encryption	Maximum Clients
SSIDs Fortinet Recommended - Factory Default (3)							
□	Corporate_Fortinet_Default	Corporate	Local Bridge	WPA2 Enterprise	Always	AES	0
□	Guest_Fortinet_Default	Guest	Tunnel	Captive Portal	Always		0
□	POS_Fortinet_Default	POS	Local Bridge	WPA2 Personal	Always	AES	0
SSIDs (1)							
□	ss11	ss11	Tunnel	WPA2 Personal	Always	AES	0
SSID Groups (0)							

b. Right click on a recommended SSID and click *View* to view its details.



	Name	SSID	Traffic Mode	Security Mode	Schedule	Data Encryption	Maximum Clients
	SSIDs Fortinet Recommended - Factory Default (3)						
☑	Corporate_Fortinet_Default	Corporate	Local Bridge	WPA2 Enterprise	Always	AES	0
□	Guest_Fortinet_Default	Guest	Tunnel	Captive Portal	Always		0
□	POS_Fortinet_Default	POS	Local Bridge	WPA2 Personal	Always	AES	0
	SSIDs (1)						
□	ss11	ss11	Tunnel	WPA2 Personal	Always	AES	0
	SSID Groups (0)						



View SSID

Name: Corporate_Fortinet_Default

Alias:

Traffic Mode: Bridge

WiFi Settings

SSID: Corporate

Security Mode: WPA2 Enterprise

PMF: Disable [✓ Enable] Optional

Local Standalone

Local Authentication

Client Limit

Authentication: Local [✓ RADIUS Server]

Click to select

Broadcast SSID

Dynamic VLAN Assignment

Schedule: always Start:00:00-End:00:00 SMTWTFS

1 entry selected

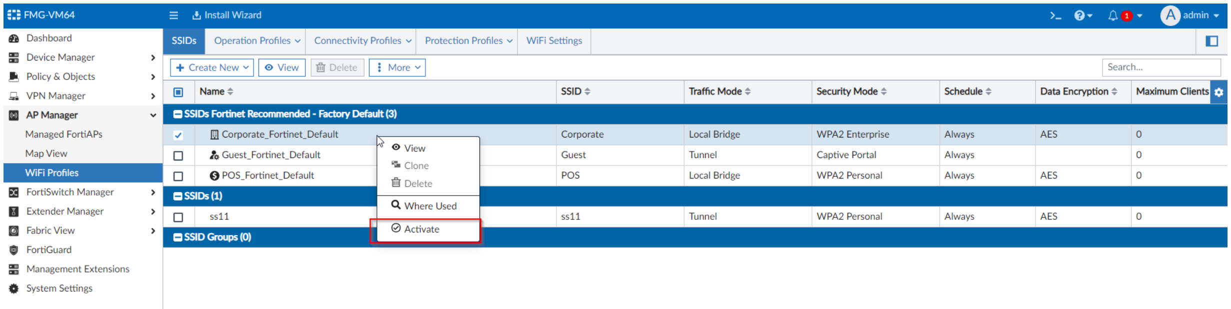
Block Intra-SSID Traffic

Optional VLAN ID: 0

Broadcast Suppression

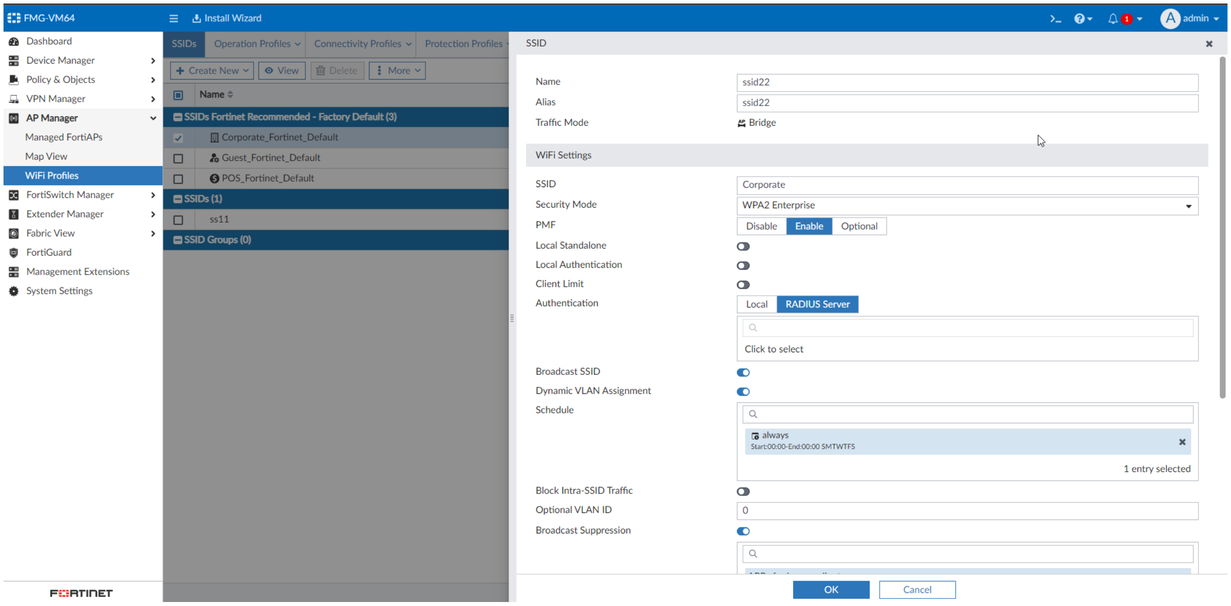
4. An SSID can be created by activating the recommended SSIDs.

a. Right-click on a recommended SSID and click *Activate*.



	Name	SSID	Traffic Mode	Security Mode	Schedule	Data Encryption	Maximum Clients
	SSIDs Fortinet Recommended - Factory Default (3)						
☑	Corporate_Fortinet_Default	Corporate	Local Bridge	WPA2 Enterprise	Always	AES	0
□	Guest_Fortinet_Default	Guest	Tunnel	Captive Portal	Always		0
□	POS_Fortinet_Default	POS	Local Bridge	WPA2 Personal	Always	AES	0
	SSIDs (1)						
□	ss11	ss11	Tunnel	WPA2 Personal	Always	AES	0
	SSID Groups (0)						

b. Enter a name for the SSID and configure the remaining settings as needed.



	Name
	SSIDs Fortinet Recommended - Factory Default (3)
☑	Corporate_Fortinet_Default
☐	Guest_Fortinet_Default
☐	POS_Fortinet_Default
	SSIDs (1)
☐	ss11
	SSID Groups (0)

SSID

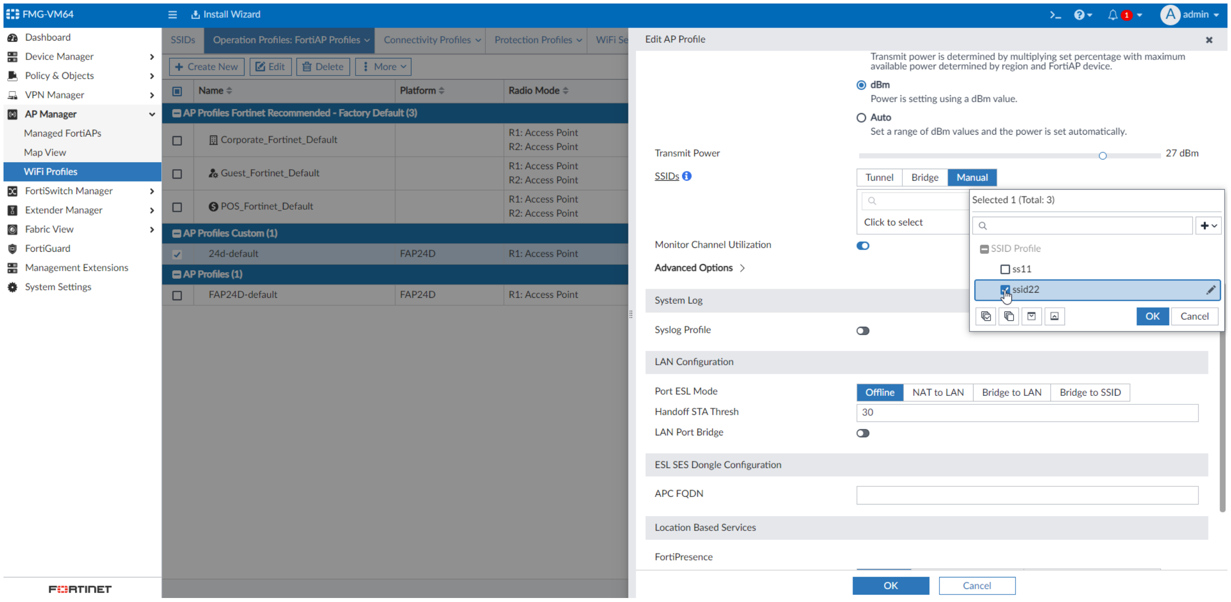
Name	ssid22
Alias	ssid22
Traffic Mode	Bridge

WiFi Settings

SSID	Corporate
Security Mode	WPA2 Enterprise
PMF	Disable [✓ Enable] Optional
Local Standalone	off
Local Authentication	off
Client Limit	off
Authentication	Local [✓ RADIUS Server] Click to select
Broadcast SSID	on
Dynamic VLAN Assignment	on
Schedule	always Start:00:00-End:00:00 SMTWTFS 1 entry selected
Block Intra-SSID Traffic	off
Optional VLAN ID	0
Broadcast Suppression	on

5. The created SSID can be assigned to an AP profile, and the profile can be assigned to the FortiAP.

a. In the FortiAP Profile, select the configured SSID.



	Name	Platform	Radio Mode
	AP Profiles Fortinet Recommended - Factory Default (3)		
☐	Corporate_Fortinet_Default		R1: Access Point R2: Access Point
☐	Guest_Fortinet_Default		R1: Access Point R2: Access Point
☐	POS_Fortinet_Default		R1: Access Point R2: Access Point
	AP Profiles Custom (1)		
☑	24d-default	FAP24D	R1: Access Point
	AP Profiles (1)		
☐	FAP24D-default	FAP24D	R1: Access Point

Edit AP Profile

Transmit power is determined by multiplying set percentage with maximum available power determined by region and FortiAP device.

● dBm
Power is setting using a dBm value.

○ Auto
Set a range of dBm values and the power is set automatically.

Transmit Power	27 dBm
SSIDs	Tunnel Bridge [✓ Manual] Click to select
Monitor Channel Utilization	on

Selected 1 (Total: 3)

SSID Profile
- ☐ ss11
- ☑ ssid22

Advanced Options

System Log

Syslog Profile	off

LAN Configuration

Port ESL Mode	[✓ Offline] NAT to LAN Bridge to LAN Bridge to SSID
Handoff STA Thresh	30
LAN Port Bridge	off

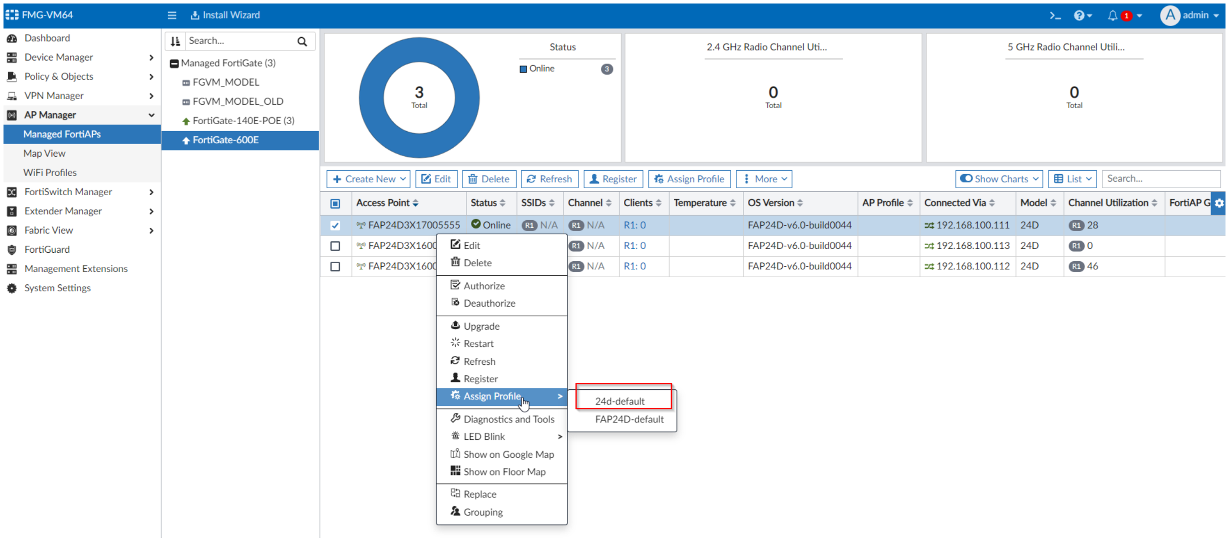
ESL SES Dongle Configuration

APC FQDN	

Location Based Services

FortiPresence

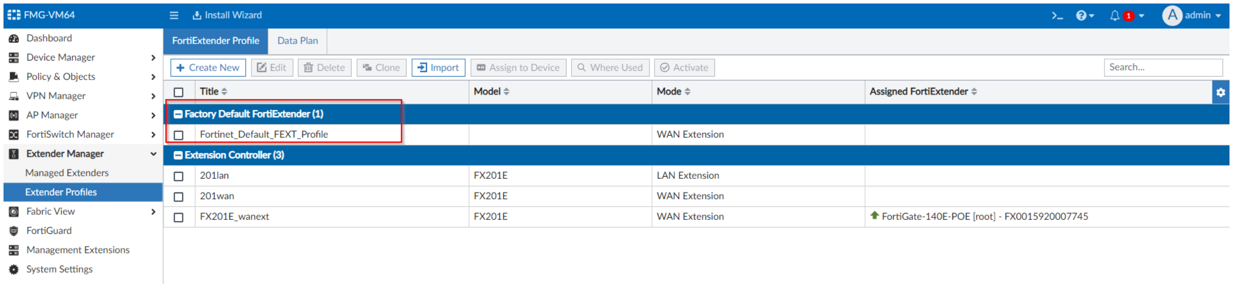
b. Assign the profile to the FortiAP device.



	Access Point	Status	SSIDs	Channel	Clients	Temperature	OS Version	AP Profile	Connected Via	Model	Channel Utilization	FortiAP G
☑	FAP24D3X17005555	Online	R1 N/A	R1 N/A	R1: 0		FAP24D-v6.0-build0044		192.168.100.111	24D	R1 28	
□	FAP24D3X1600			R1 N/A	R1: 0		FAP24D-v6.0-build0044		192.168.100.113	24D	R1 0	
□	FAP24D3X1600			R1 N/A	R1: 0		FAP24D-v6.0-build0044		192.168.100.112	24D	R1 46	

To use recommended FortiExtender templates:

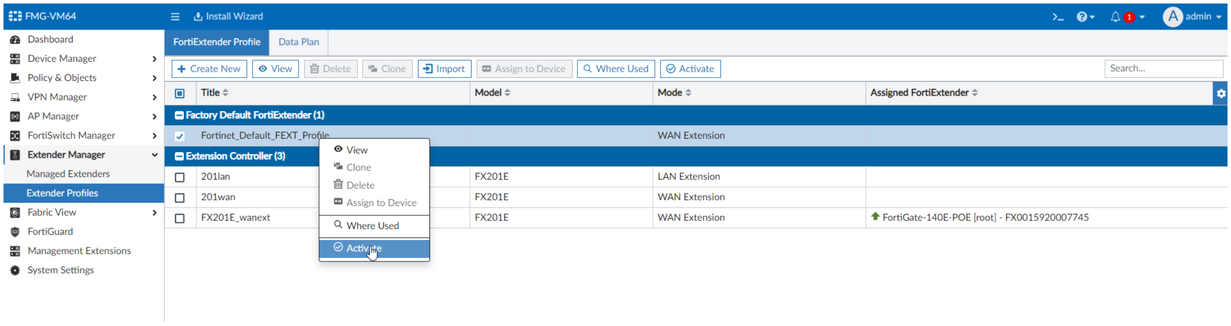
1. The recommended Extender Profile is shown in *Extender Manager* on the *Extender Profiles* tab.



□	Title	Model	Mode	Assigned FortiExtender
	Factory Default FortiExtender (1)			
□	Fortinet_Default_FEXT_Profile		WAN Extension	
	Extension Controller (3)			
□	201lan	FX201E	LAN Extension	
□	201wan	FX201E	WAN Extension	
□	FX201E_wanext	FX201E	WAN Extension	FortiGate-140E-POE [root] - FX0015920007745

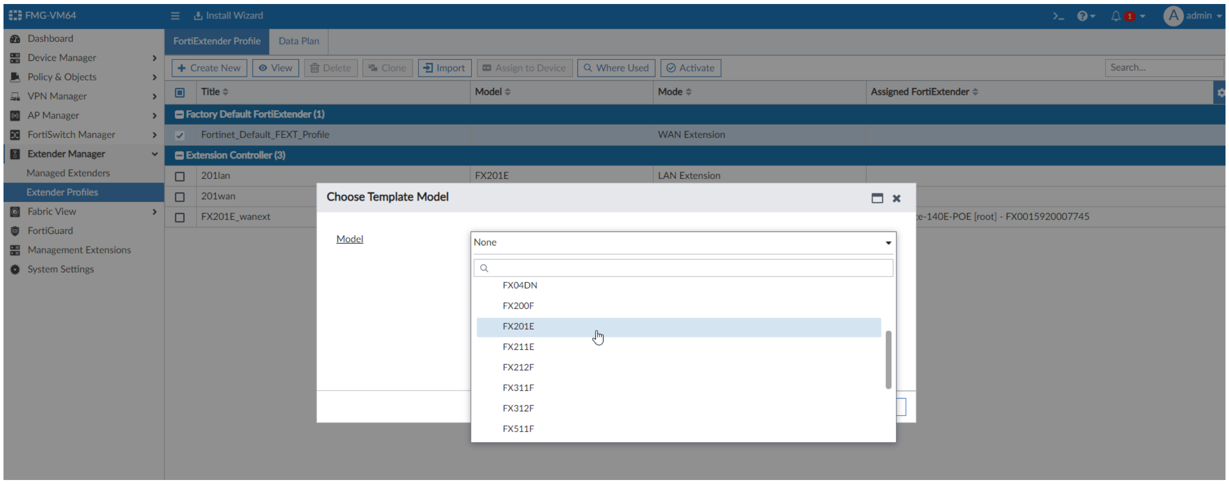
2. An extender profile can be created by activating the recommended FortiExtender profile.

   a. Right-click on the recommended FortiExtender profile and click *Activate*.

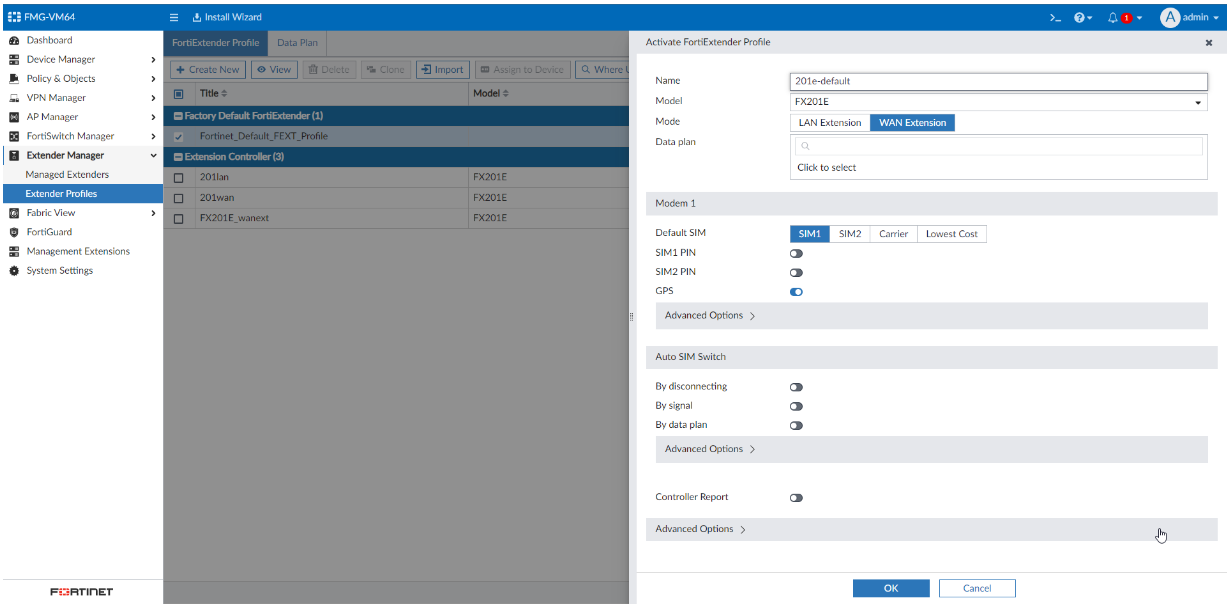


□	Title	Model	Mode	Assigned FortiExtender
	Factory Default FortiExtender (1)			
☑	Fortinet_Default_FEXT_Profile		WAN Extension	
	Extension Controller (3)			
□	201lan	FX201E	LAN Extension	
□	201wan	FX201E	WAN Extension	
□	FX201E_wanext	FX201E	WAN Extension	FortiGate-140E-POE [root] - FX0015920007745

b. Choose a model for the template.



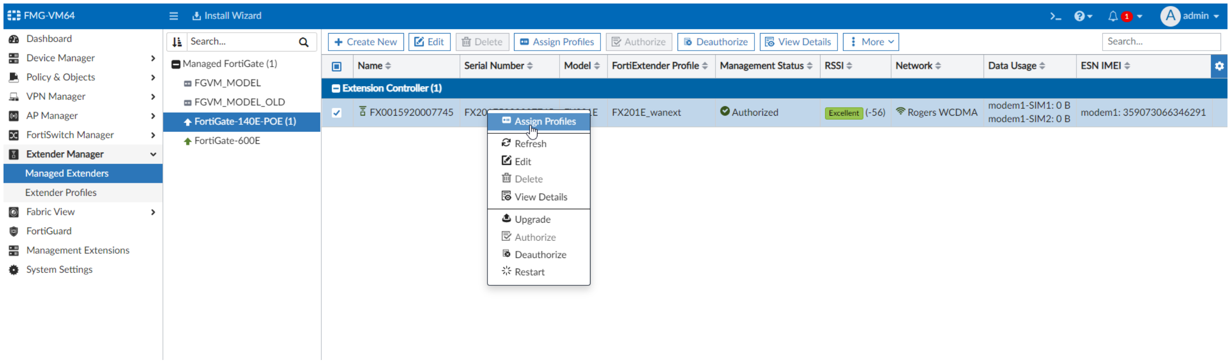
c. Enter a name for the FortiExtender profile and configure the remaining settings as needed.



Name: 201e-default
Model: FX201E
Mode: LAN Extension [✓ WAN Extension]
Default SIM: [✓ SIM1] SIM2 Carrier Lowest Cost
GPS: ●

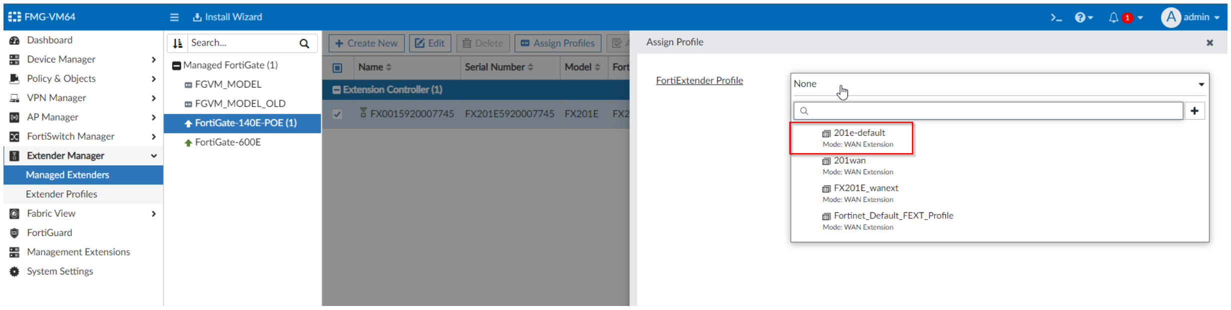
3. The created extender profile can be assigned to an extender, then the user can deploy the settings.

a. Right-click on a managed FortiExtender and click *Assign Profiles*.



Name	Serial Number	Model	FortiExtender Profile	Management Status	RSSI	Network	Data Usage	ESN IMEI
Extension Controller (1)								
FX0015920007745	FX20…	…E	FX201E_wanext	Authorized	Excellent (-56)	Rogers WCDMA	modem1-SIM1: 0 B modem1-SIM2: 0 B	modem1: 359073066346291

b. Select the configured FortiExtender Profile.



SD-WAN template for heterogeneous WAN link types - FMG

This information is also available in the FortiManager 7.4 Administration Guide:

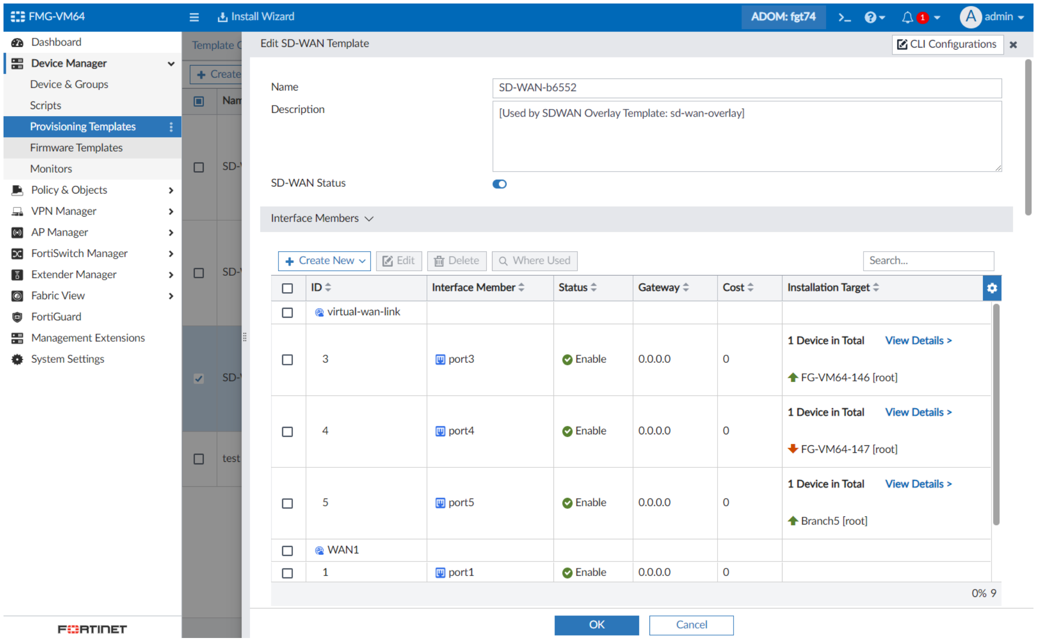
- Zones and interface members
- Performance SLA

SD-WAN template for heterogeneous WAN link types to support single-template usage for FortiGates with different underlay connections and SLAs.

To create an SD-WAN template with heterogeneous WAN links:

1. Go to *Device Manager* > *Provisioning Templates* > *SD-WAN Template*, and create or edit a template.
2. You can select the installation target for the following SD-WAN objects:
   - system sdwan members
   - system sdwan service
   - system sdwan health-check

- system sdwan neighbor



Edit SD-WAN Template

Name: SD-WAN-b6552

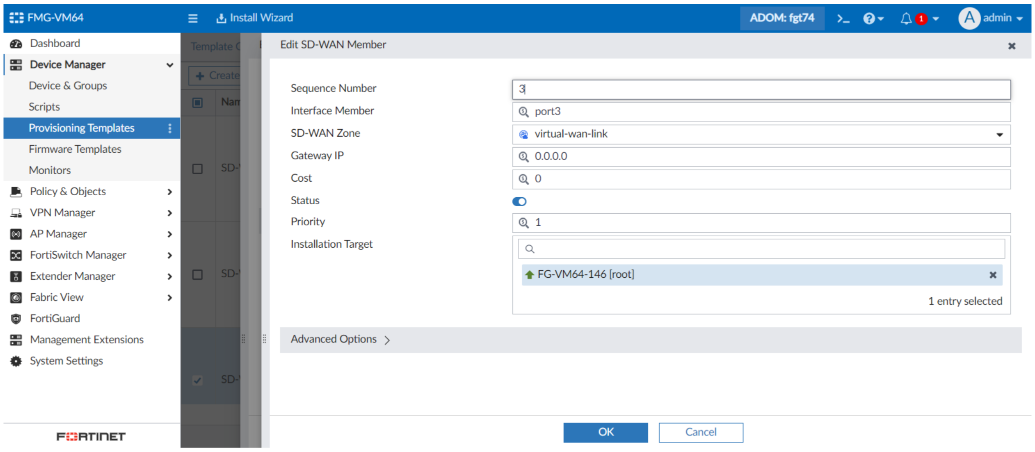
Description: [Used by SDWAN Overlay Template: sd-wan-overlay]

SD-WAN Status: ●

Interface Members

ID	Interface Member	Status	Gateway	Cost	Installation Target
virtual-wan-link					
3	port3	Enable	0.0.0.0	0	1 Device in Total View Details > FG-VM64-146 [root]
4	port4	Enable	0.0.0.0	0	1 Device in Total View Details > FG-VM64-147 [root]
5	port5	Enable	0.0.0.0	0	1 Device in Total View Details > Branch5 [root]
WAN1					
1	port1	Enable	0.0.0.0	0	

0% 9



Edit SD-WAN Member

Sequence Number: 3

Interface Member: port3

SD-WAN Zone: virtual-wan-link

Gateway IP: 0.0.0.0

Cost: 0

Status: ●

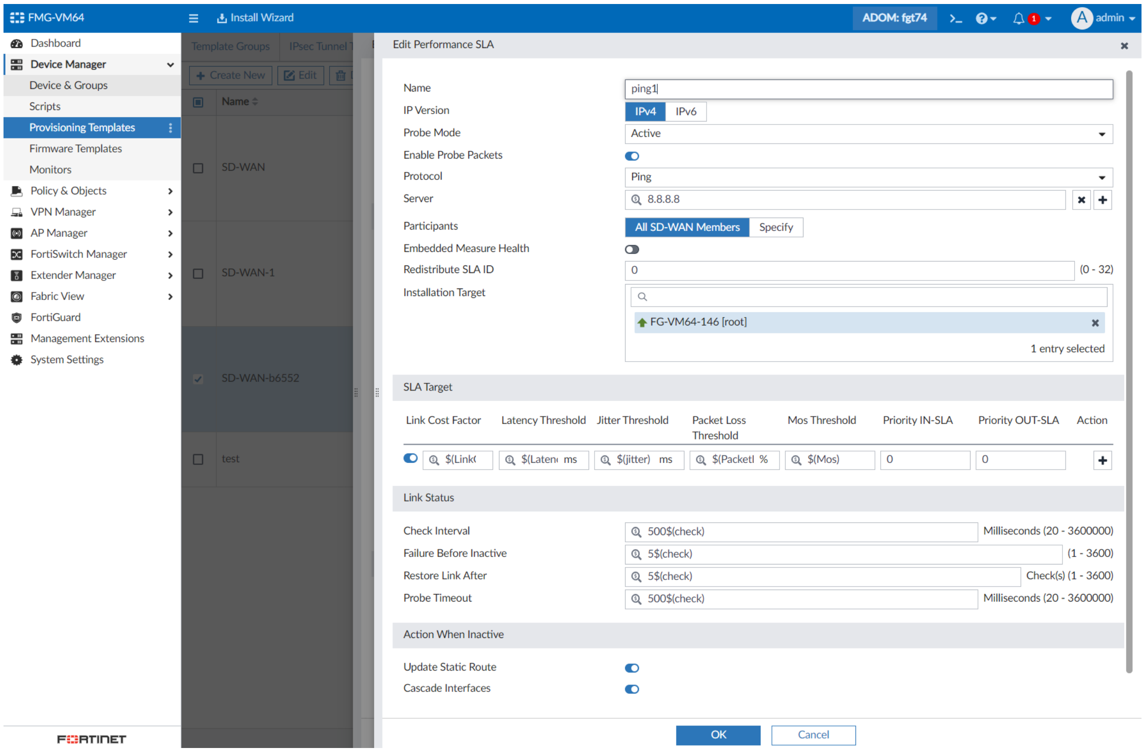
Priority: 1

Installation Target: FG-VM64-146 [root]

1 entry selected

3. You can add meta variables for the following health-check attributes:
   - System SD-WAN health-check
     - Check Interval
     - Fail Before Inactive
     - Probe Timeout
     - Restore Link After
   - System SD-WAN health-check SLA
     - Link Cost Factor
     - Latency Threshold
     - Jitter Threshold
     - Packet Loss Threshold

- Mos Threshold



Edit Performance SLA

Name	ping1
IP Version	IPv4 / IPv6
Probe Mode	Active
Enable Probe Packets	●
Protocol	Ping
Server	8.8.8.8
Participants	All SD-WAN Members / Specify
Embedded Measure Health	○
Redistribute SLA ID	0 (0 - 32)
Installation Target	FG-VM64-146 [root] 1 entry selected

SLA Target

Link Cost Factor	Latency Threshold	Jitter Threshold	Packet Loss Threshold	Mos Threshold	Priority IN-SLA	Priority OUT-SLA	Action
$(Link(	$(Laten(ms	$(jitter) ms	$(Packetl %	$(Mos)	0	0	+

Link Status

Check Interval	500$(check) Milliseconds (20 - 3600000)
Failure Before Inactive	5$(check) (1 - 3600)
Restore Link After	5$(check) Check(s) (1 - 3600)
Probe Timeout	500$(check) Milliseconds (20 - 3600000)

Action When Inactive

Update Static Route	●
Cascade Interfaces	●

OK Cancel

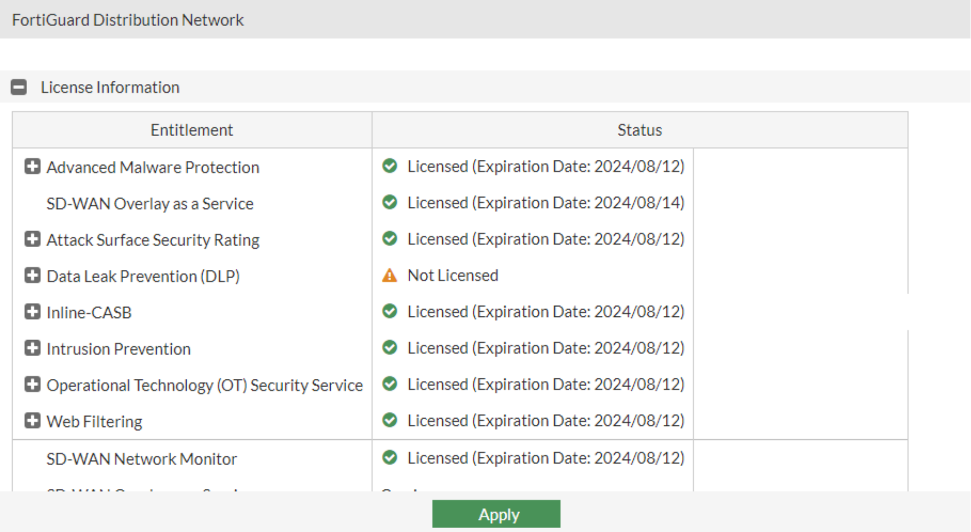
Support the new SD-WAN Overlay-as-a-Service - 7.4.1

SD-WAN Overlay-as-a-Service (OaaS) is supported through a license displayed as *SD-WAN Overlay as a Service* on the *System* > *FortiGuard* page. Each FortiGate used by the FortiCloud Overlay-as-a-Service portal must have this license applied to it.

To view the status of the OaaS license in the GUI:

1. Go to *System* > *FortiGuard*.
2. Expand *License Information*. The *SD-WAN Overlay as a Service* license status is listed as:

- *Licensed*: OaaS is currently licensed and will expire on the provided date.

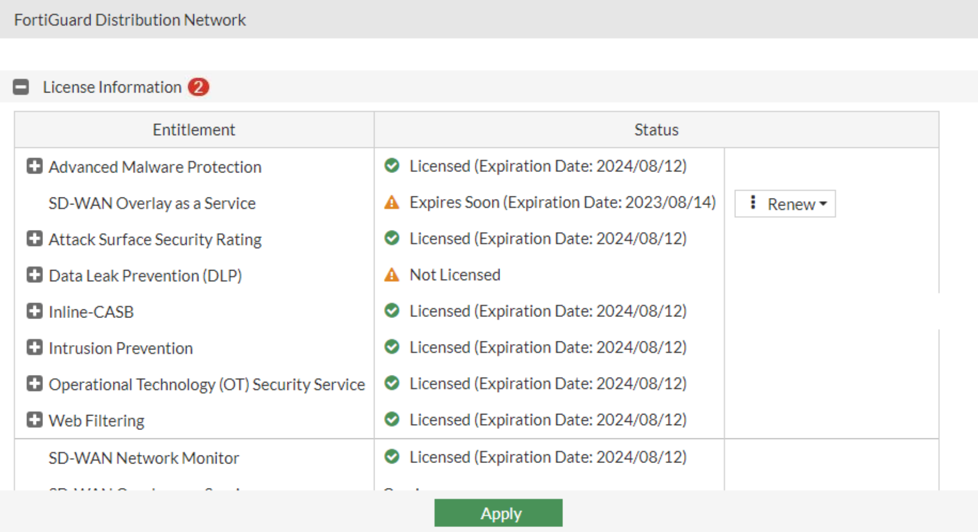


FortiGuard Distribution Network

License Information

Entitlement	Status
Advanced Malware Protection	✓ Licensed (Expiration Date: 2024/08/12)
SD-WAN Overlay as a Service	✓ Licensed (Expiration Date: 2024/08/14)
Attack Surface Security Rating	✓ Licensed (Expiration Date: 2024/08/12)
Data Leak Prevention (DLP)	⚠ Not Licensed
Inline-CASB	✓ Licensed (Expiration Date: 2024/08/12)
Intrusion Prevention	✓ Licensed (Expiration Date: 2024/08/12)
Operational Technology (OT) Security Service	✓ Licensed (Expiration Date: 2024/08/12)
Web Filtering	✓ Licensed (Expiration Date: 2024/08/12)
SD-WAN Network Monitor	✓ Licensed (Expiration Date: 2024/08/12)

- *Expires Soon*: OaaS is currently licensed but will expire soon on the provided date.

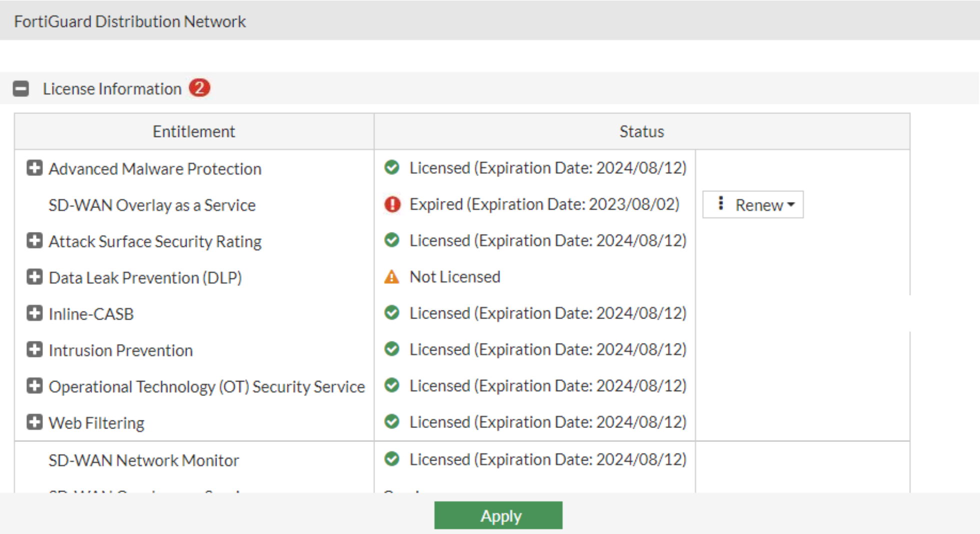


FortiGuard Distribution Network

License Information 2

Entitlement	Status	
Advanced Malware Protection	✓ Licensed (Expiration Date: 2024/08/12)	
SD-WAN Overlay as a Service	⚠ Expires Soon (Expiration Date: 2023/08/14)	⋮ Renew ▾
Attack Surface Security Rating	✓ Licensed (Expiration Date: 2024/08/12)	
Data Leak Prevention (DLP)	⚠ Not Licensed	
Inline-CASB	✓ Licensed (Expiration Date: 2024/08/12)	
Intrusion Prevention	✓ Licensed (Expiration Date: 2024/08/12)	
Operational Technology (OT) Security Service	✓ Licensed (Expiration Date: 2024/08/12)	
Web Filtering	✓ Licensed (Expiration Date: 2024/08/12)	
SD-WAN Network Monitor	✓ Licensed (Expiration Date: 2024/08/12)	

- *Expired*: The OaaS license has already expired on the provided date.



FortiGuard Distribution Network

⊟ License Information 2

Entitlement	Status	
⊞ Advanced Malware Protection	✓ Licensed (Expiration Date: 2024/08/12)	
SD-WAN Overlay as a Service	! Expired (Expiration Date: 2023/08/02)	⋮ Renew ▾
⊞ Attack Surface Security Rating	✓ Licensed (Expiration Date: 2024/08/12)	
⊞ Data Leak Prevention (DLP)	⚠ Not Licensed	
⊞ Inline-CASB	✓ Licensed (Expiration Date: 2024/08/12)	
⊞ Intrusion Prevention	✓ Licensed (Expiration Date: 2024/08/12)	
⊞ Operational Technology (OT) Security Service	✓ Licensed (Expiration Date: 2024/08/12)	
⊞ Web Filtering	✓ Licensed (Expiration Date: 2024/08/12)	
SD-WAN Network Monitor	✓ Licensed (Expiration Date: 2024/08/12)	

To view the status of the OaaS license in the CLI:

1. Verify that the entitlement can be updated:

The SD-WAN Overlay-as-a-Service license is listed as `SWOS` in the CLI.

```
# diagnose test update info

System contracts:
    FMWR,Wed Dec 20 16:00:00 2023
    SPAM,Wed Dec 20 16:00:00 2023
    SBCL,Wed Dec 20 16:00:00 2023
    SWNO,Wed Dec 20 16:00:00 2023
    SWNM,Wed Sep 27 17:00:00 2023
    SWOS,Mon Aug 14 17:00:00 2023
    SPRT,Wed Dec 20 16:00:00 2023
    SDWN,Sun Dec 10 16:00:00 2023
    SBCL,Wed Dec 20 16:00:00 2023
    SBEN,Wed Dec 20 16:00:00 2023
```

2. Verify that the expiration date log can be generated:

```
# execute log display

1: date=2023-08-10 time=00:00:01 eventtime=1691650800645347120 tz="-0700"
logid="0100020138" type="event" subtype="system" level="warning" vd="root"
logdesc="FortiGuard SD-WAN Overlay as a Service license expiring" msg="FortiGuard SD-WAN
Overlay Service license will expire in 4 day(s)"
```

Fabric Authorization Template is integrated with Device Blueprint and supports meta variables - FMG 7.4.1

This information is also available in the FortiManager 7.4 Administration Guide:

- Using Device Blueprints for Model Devices
- Fabric Authorization Templates

In FortiManager 7.4.1, the Fabric Authorization Template is integrated with Device Blueprints and supports metadata variables.

To use Fabric Authorization Templates with Device Blueprints:

- Step 1: Configure the Fabric Authorization Template on page 97
- Step 2: Add a Fabric Authorization Template to a device blueprint on page 98
- Step 3: Add the device blueprint to a model device on page 99
- Step 4: View the configured FortiAP, FortiSwitch, and FortiExtender on page 99

Step 1: Configure the Fabric Authorization Template

To create a Fabric Authorization Template:

1. Go to *Device Manager* > *Provisioning Templates* > *Fabric Authorization.*
2. Click *Create New*, and the *Create New Fabric Authorization Template* dialog appears.
3. Configure the Fabric Authorization Template, for example a template named "60E" is configured in the with the following settings:
    a. FortiAP:

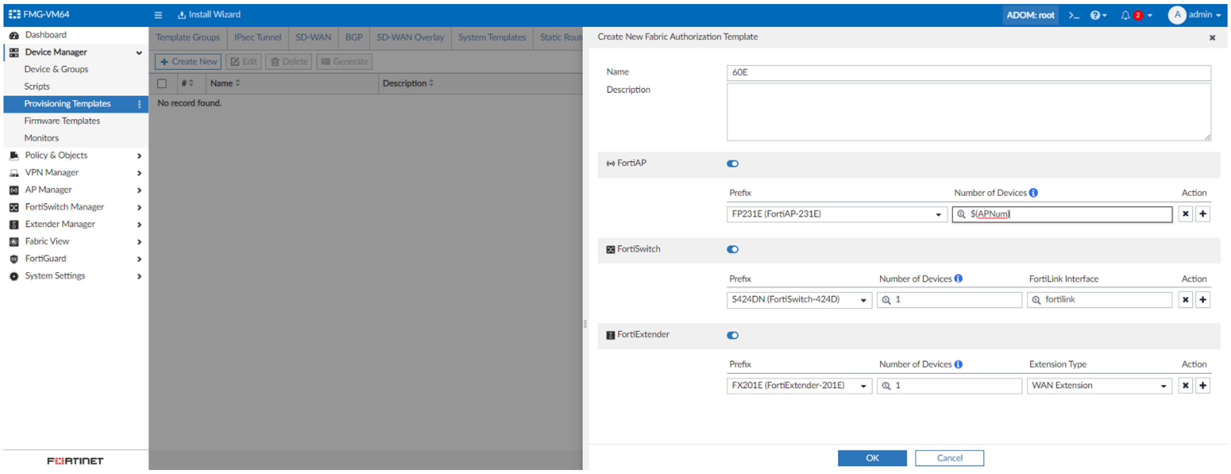
FortiAP	Toggle ON to enable FortiAP in the template.
Prefix	Select a model from the dropdown list, for example *FP231E*.
Number of Devices	This field determines how many APs will be added, and you can enter a number or use a variable. Entering the `$` sign causes the variable list to appear where you can select a variable from the dropdown list. This example uses the `$(APNum)` variable.

    b. FortiSwitch:

FortiSwitch	Toggle ON to enable FortiSwitch in the template.
Prefix	Select a model from the dropdown list, for example *S424DN*.
Number of Devices	This field determines how many FortiSwitches will be added, and you can enter a number or use a variable. This example uses the number `1`.
FortiLink Interface	Enter the interface name. This field supports variables. This example uses the name `fortilink`.

c. FortiExtender:

FortiExtender	Toggle ON to enable FortiExtender in the template.
Prefix	Select a model from the dropdown list, for example *FX201E*.
Number of Devices	This field determines how many FortiExtenders will be added, and you can enter a number or use a variable. This example uses the number `1`.
FortiLink Interface	Select an extension type from the dropdown list. This example uses *WAN Extension*.



Create New Fabric Authorization Template

Name: 60E

Description:

FortiAP: ON

Prefix	Number of Devices	Action
FP231E (FortiAP-231E)	$(APNum)	✖ ✚

FortiSwitch: ON

Prefix	Number of Devices	FortiLink Interface	Action
S424DN (FortiSwitch-424D)	1	fortilink	✖ ✚

FortiExtender: ON

Prefix	Number of Devices	Extension Type	Action
FX201E (FortiExtender-201E)	1	WAN Extension	✖ ✚

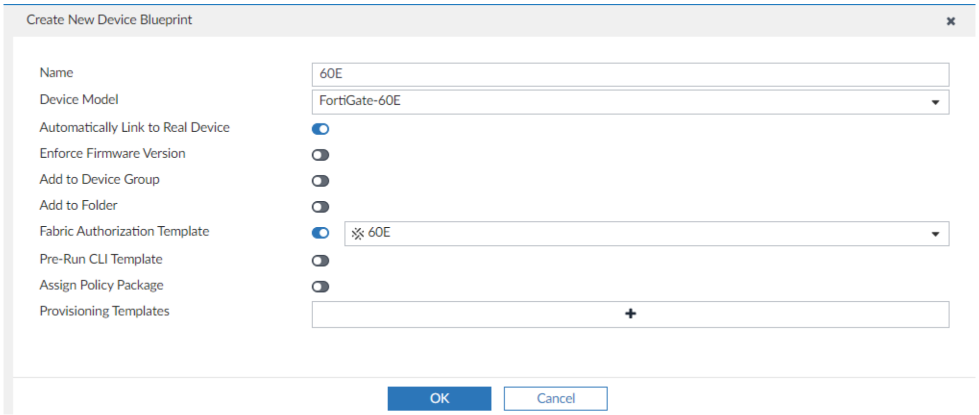
4. Click *OK* to save the Fabric Authorization Template.

Step 2: Add a Fabric Authorization Template to a device blueprint

To add the Fabric Authorization Template to a device blueprint:

1. Go to Device Manager. In the *Device & Group* window, click the *Add Device* dropdown and choose *Device Blueprint* to create a new device blueprint.
2. Configure the device blueprint. For example:

Name	Enter a name for the blueprint. In this example, it is `60E`.
Device Model	Select a device platform, for example *FortiGate-60E*.
Fabric Authorization Template	Select the previously configured Fabric Authorization Template (*60E*).



Create New Device Blueprint

Name	60E
Device Model	FortiGate-60E
Automatically Link to Real Device	●
Enforce Firmware Version	○
Add to Device Group	○
Add to Folder	○
Fabric Authorization Template	● 60E
Pre-Run CLI Template	○
Assign Policy Package	○
Provisioning Templates	+

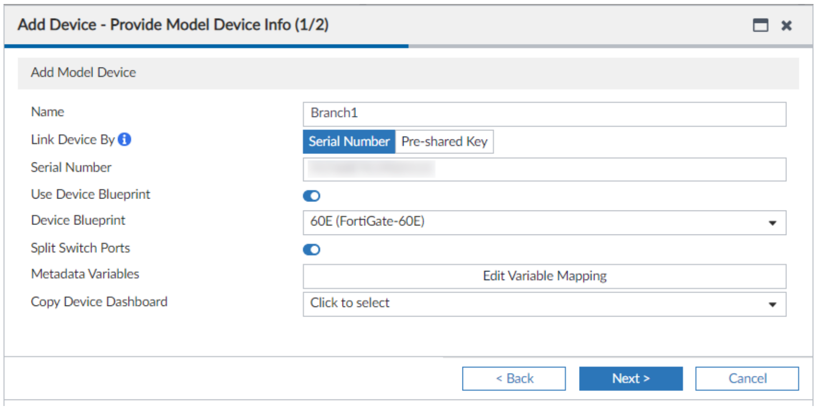
3. Click *OK* to save the device blueprint.

Step 3: Add the device blueprint to a model device

To add the Fabric Authorization Template to a model device:

1. Go to Device Manager. In the *Device & Group* window, click *Add Device* > *Add Model Device*.
2. Configure the model device using the device blueprint. For example:

Name	Enter the FortiGate's name. In this example, it is `Branch1`.
Link Device By	Select *Serial Number*.
Serial Number	Enter the serial number.
Use Device Blueprint	Enable the *Use Device Blueprint* setting.
Device Blueprint	Select the previously configured device blueprint (*`60E`*)



Add Device - Provide Model Device Info (1/2)

Add Model Device

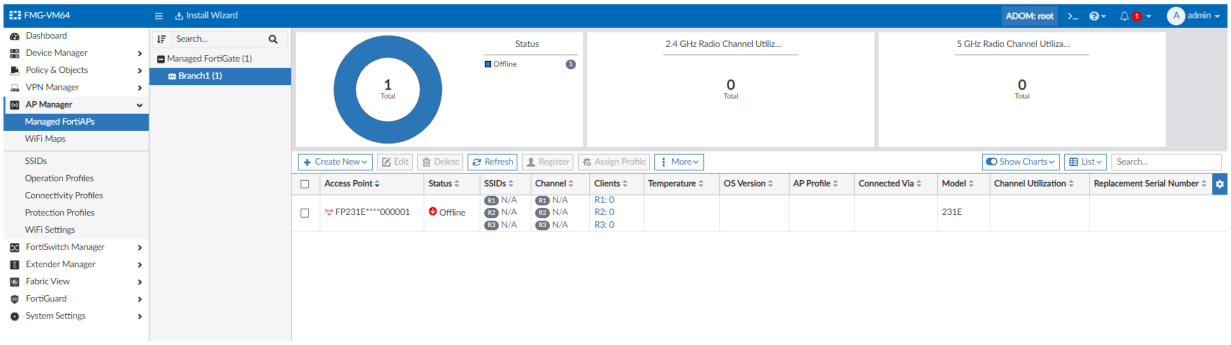
Name	Branch1
Link Device By	Serial Number / Pre-shared Key
Serial Number	[redacted]
Use Device Blueprint	●
Device Blueprint	60E (FortiGate-60E)
Split Switch Ports	●
Metadata Variables	Edit Variable Mapping
Copy Device Dashboard	Click to select

3. Click *OK* to save the model device.

Step 4: View the configured FortiAP, FortiSwitch, and FortiExtender

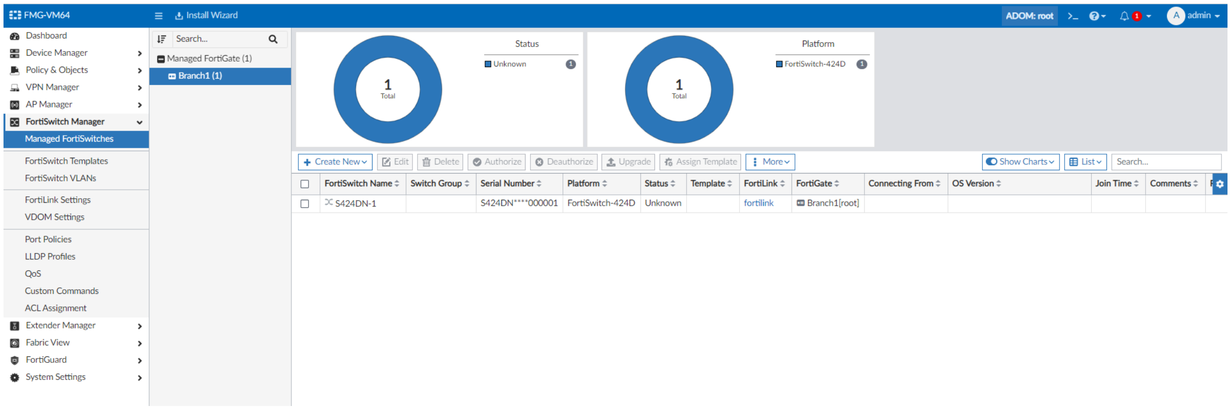
1. After the device is added to the Device Manager, go to the *AP Manager*, *FortiSwitch Manager*, and *Extender Manager* in FortiManager and you can see that the FortiGate has been automatically configured with a FortiAP, FortiSwitch and FortiExtender as defined by the template.
   For example:

- FortiAP:



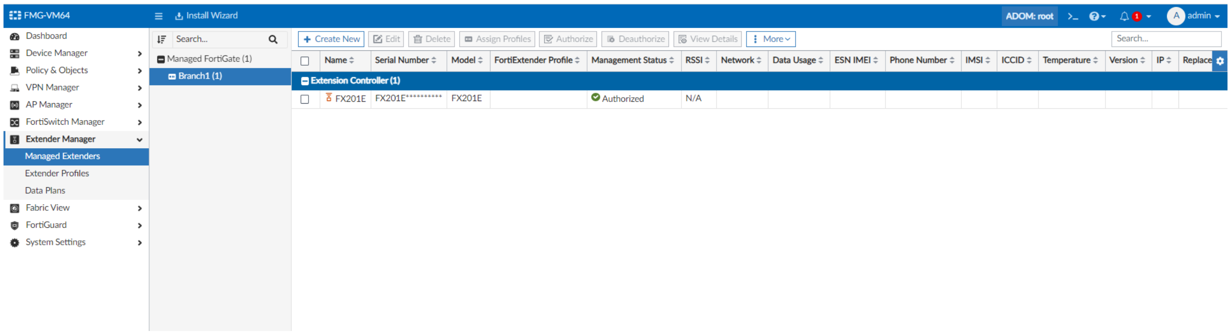
□	Access Point	Status	SSIDs	Channel	Clients	Temperature	OS Version	AP Profile	Connected Via	Model	Channel Utilization	Replacement Serial Number
□	FP231E****000001	Offline	R1 N/A R2 N/A R3 N/A	R1 N/A R2 N/A R3 N/A	R1: 0 R2: 0 R3: 0					231E		

- FortiSwitch:



□	FortiSwitch Name	Switch Group	Serial Number	Platform	Status	Template	FortiLink	FortiGate	Connecting From	OS Version	Join Time	Comments
□	S424DN-1		S424DN****000001	FortiSwitch-424D	Unknown		fortilink	Branch1[root]				

- FortiExtender:

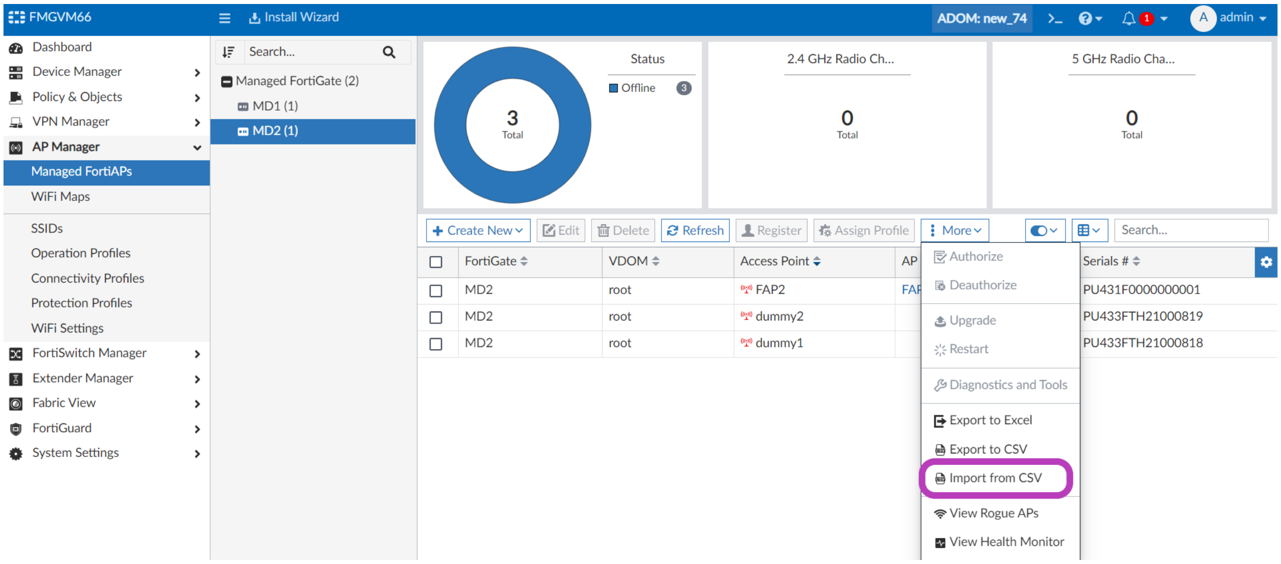


□	Name	Serial Number	Model	FortiExtender Profile	Management Status	RSSI	Network	Data Usage	ESN IMEI	Phone Number	IMSI	ICCID	Temperature	Version	IP	Replace
Extension Controller (1)																
□	FX201E	FX201E**********	FX201E		Authorized	N/A										

Export Managed FortiAPs and import FortiAPs from a CSV file - FMG 7.4.2

To import and export managed FortiAPs:

1. Go to *AP Manager* > *Managed FortiAPs*, and click *More* > *Import from CSV*. Users can import AP tables as a CSV file.

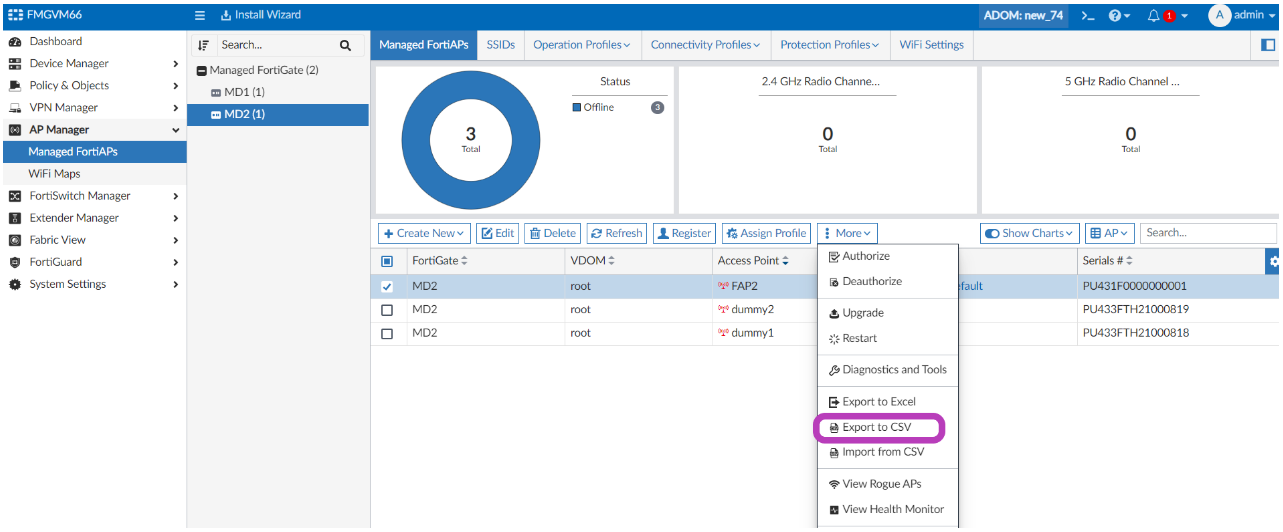


FortiGate	VDOM	Access Point	AP	Serials #
MD2	root	FAP2	FAP	PU431F0000000001
MD2	root	dummy2		PU433FTH21000819
MD2	root	dummy1		PU433FTH21000818

The following columns can be imported from the CSV:

Column Header	Cell Information
FortiGate	The name of the FortiGate to which the FortiAP will be assigned.
Access Point	The access point name.
Serial #	The FortiAP serial number
FAP Profile	The profile to be assigned to the FortiAP.
VDOM name	(Optional) If VDOMs are enabled on the FortiGate, specify the VDOM to which the FortiAP will be assigned. If VDOMs are disabled, leave

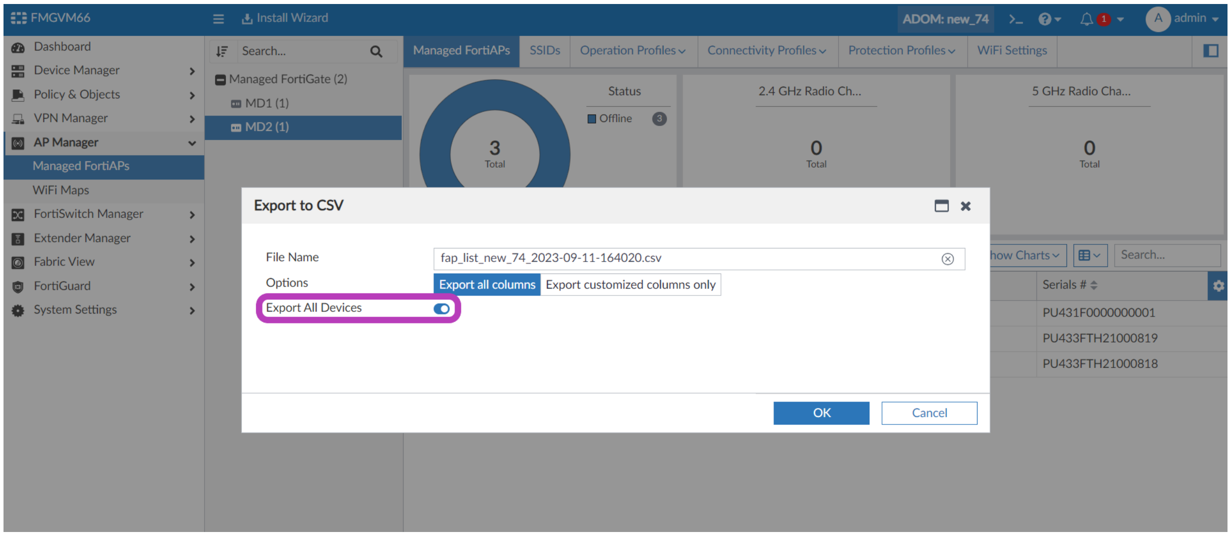
2. Go to *AP Manager* > *Managed FortiAPs*, and click *More* > *Export to CSV*.



	FortiGate	VDOM	Access Point		Serials #
☑	MD2	root	FAP2	fault	PU431F0000000001
□	MD2	root	dummy2		PU433FTH21000819
□	MD2	root	dummy1		PU433FTH21000818

3. Configure the export options:
   - *File Name*: Enter the name of the file.
   - *Options*: Select *Export all columns* or *Export customized columns only*.

- *Export All Devices*: This option is displayed in per-device mode only. You can enable this toggle to include all managed FortiGate's FortiAPs in the CSV file.



Export to CSV

File Name: fap_list_new_74_2023-09-11-164020.csv

Options: [✓ Export all columns] Export customized columns only

Export All Devices: ● (enabled)

Meta variables are available in the SSID, FortiSwitch VLANs, and FortiSwitch Templates configuration - FMG 7.4.2

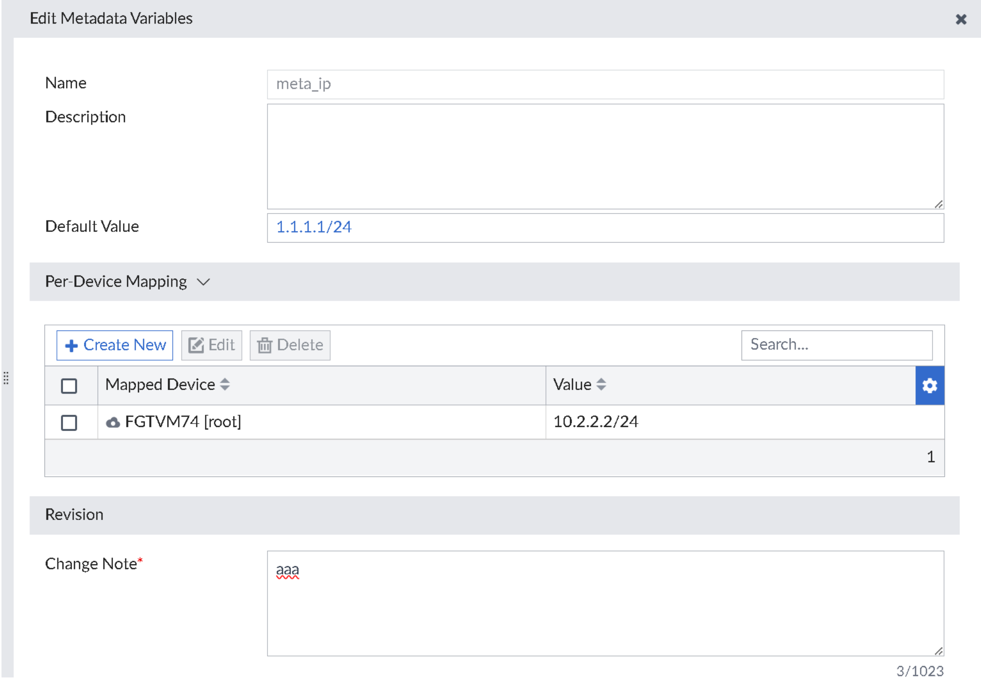
This information is also available in the FortiManager 7.4 Administration Guide:

- ADOM-level metadata variables

Meta variables are available in the SSID, FortiSwitch VLANs, and FortiSwitch Templates configuration

To use metadata variables in AP Manager and FortiSwitch Manager:

1. Go to *Policy & Objects* > *Advanced* > *Metadata Variables*, and create a new metadata variable.



Name	meta_ip
Description	
Default Value	1.1.1.1/24

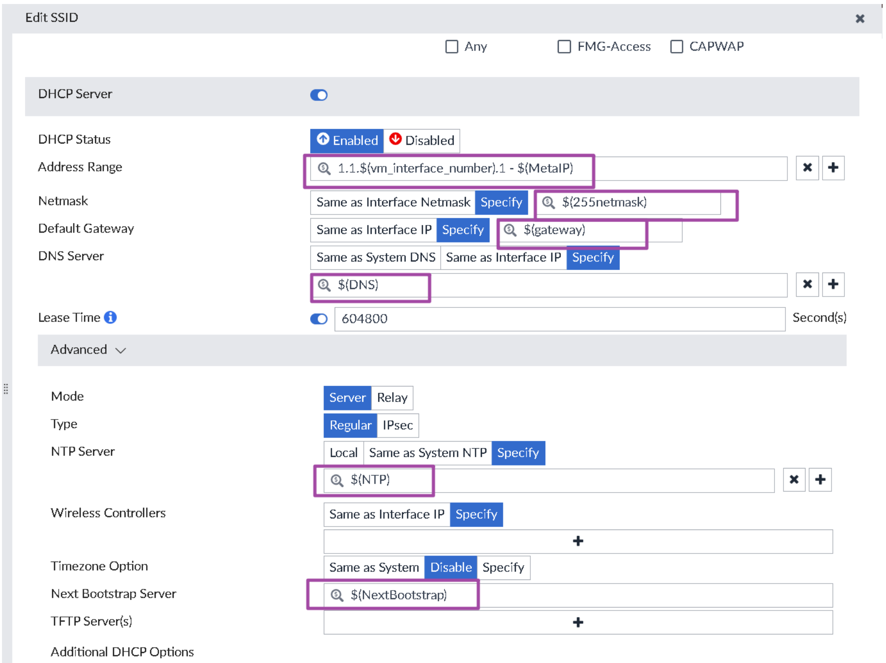
□	Mapped Device	Value
□	FGTVM74 [root]	10.2.2.2/24
		1

Change Note*	aaa

3/1023

2. Go to *AP Manager* > *SSID* and create or edit an SSID.
   Fields which support metadata variables contain a magnifying glass icon. These fields include:
   - IP/Network Mask
   - DHCP Server Address Range
   - Netmask
   - Default Gateway
   - DNS Server
   - NTP Server

- Next Boostrap Server



Edit SSID

□ Any □ FMG-Access □ CAPWAP

DHCP Server

DHCP Status: Enabled

Address Range: 1.1.${vm_interface_number}.1 - ${MetalP}

Netmask: Specify ${255netmask}

Default Gateway: Specify ${gateway}

DNS Server: Specify ${DNS}

Lease Time: 604800 Second(s)

Advanced

Mode: Server

Type: Regular

NTP Server: Specify ${NTP}

Wireless Controllers: Specify

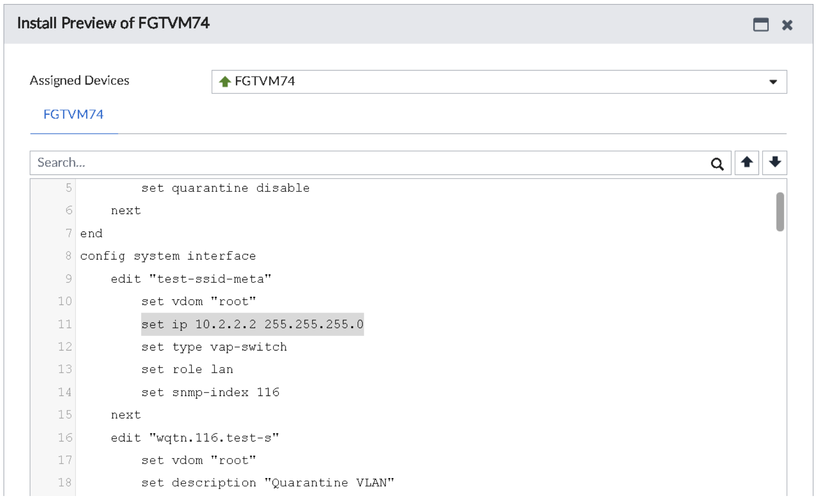
Timezone Option: Disable

Next Bootstrap Server: ${NextBootstrap}

TFTP Server(s)

Additional DHCP Options

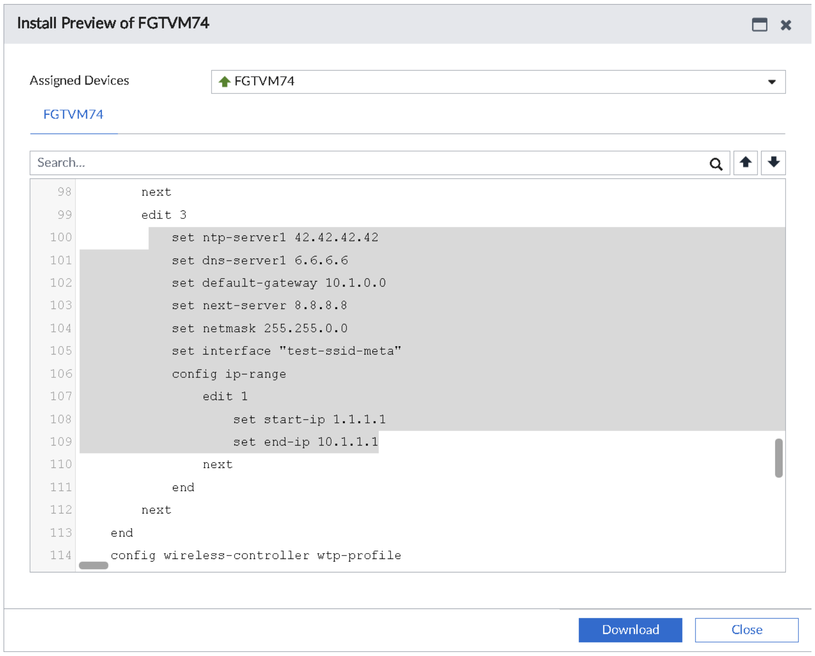
3. When reviewing the *Install Preview* of an SSID which contains metadata variables, the preview displays the resolved previews.



Install Preview of FGTVM74

Assigned Devices: FGTVM74

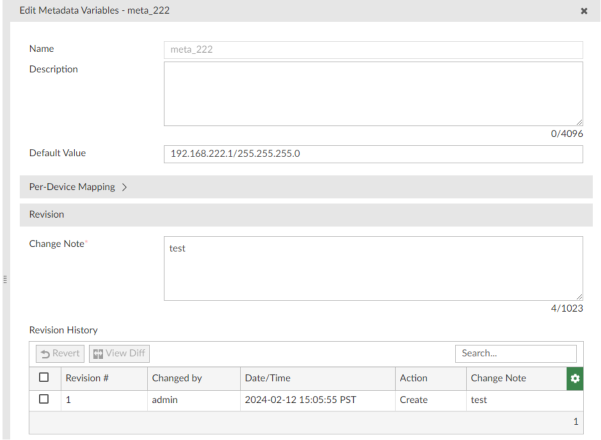
```
        set quarantine disable
    next
end
config system interface
    edit "test-ssid-meta"
        set vdom "root"
        set ip 10.2.2.2 255.255.255.0
        set type vap-switch
        set role lan
        set snmp-index 116
    next
    edit "wqtn.116.test-s"
        set vdom "root"
        set description "Quarantine VLAN"
```



```
        next
        edit 3
            set ntp-server1 42.42.42.42
            set dns-server1 6.6.6.6
            set default-gateway 10.1.0.0
            set next-server 8.8.8.8
            set netmask 255.255.0.0
            set interface "test-ssid-meta"
            config ip-range
                edit 1
                    set start-ip 1.1.1.1
                    set end-ip 10.1.1.1
                next
            end
        next
    end
    config wireless-controller wtp-profile
```

To use metadata variables in FortiSwitch Manager:

1. Go to *Policy & Objects* > *Advanced* > *Metadata Variables*, and create a new metadata variable.

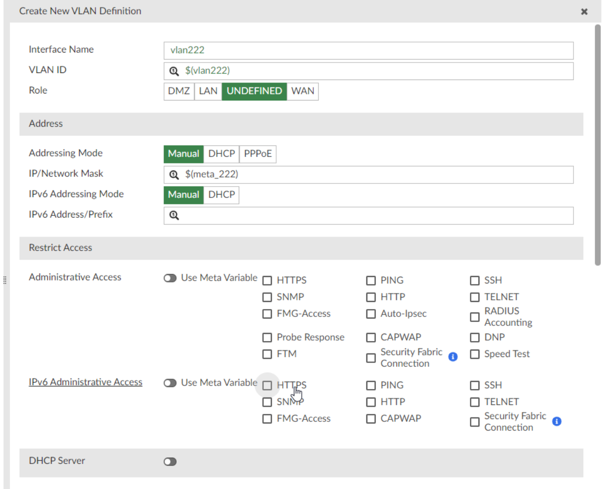


Name	meta_222
Description	
	0/4096
Default Value	192.168.222.1/255.255.255.0
Change Note*	test
	4/1023

☐	Revision #	Changed by	Date/Time	Action	Change Note
☐	1	admin	2024-02-12 15:05:55 PST	Create	test

1

2. Go to *FortiSwitch Manager* > *FortiSwitch VLANs*. Metadata variables can be used in FortiSwitch VLANs.



Create New VLAN Definition

Interface Name: vlan222
VLAN ID: $(vlan222)
Role: UNDEFINED

Address

Addressing Mode: Manual
IP/Network Mask: $(meta_222)
IPv6 Addressing Mode: Manual
IPv6 Address/Prefix:

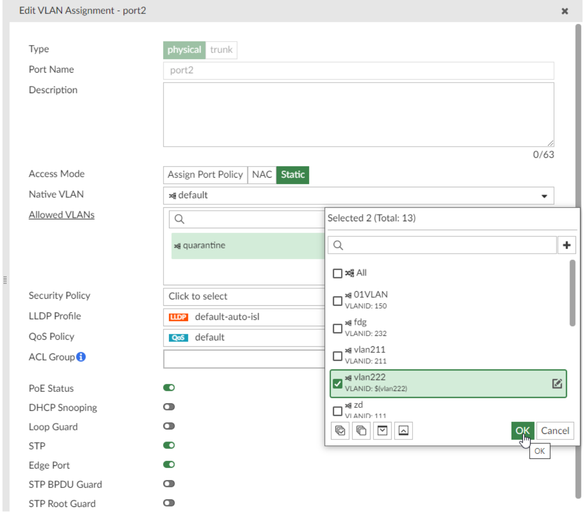
Restrict Access

Administrative Access: Use Meta Variable ○ — □ HTTPS □ PING □ SSH □ SNMP □ HTTP □ TELNET □ FMG-Access □ Auto-Ipsec □ RADIUS Accounting □ Probe Response □ CAPWAP □ DNP □ FTM □ Security Fabric Connection □ Speed Test

IPv6 Administrative Access: Use Meta Variable ○ — □ HTTPS □ PING □ SSH □ SNMP □ HTTP □ TELNET □ FMG-Access □ CAPWAP □ Security Fabric Connection

DHCP Server ○

3. After assigning the VLAN to a FortiSwitch template, the mapped values are installed.

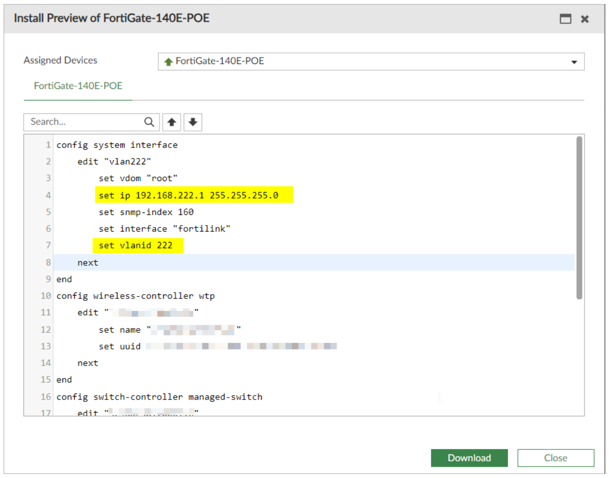


Edit VLAN Assignment - port2

Type: physical
Port Name: port2
Description:
0/63
Access Mode: Static
Native VLAN: default
Allowed VLANs: quarantine
Security Policy: Click to select
LLDP Profile: default-auto-isl
QoS Policy: default
ACL Group
PoE Status ●
DHCP Snooping ○
Loop Guard ○
STP ●
Edge Port ●
STP BPDU Guard ○
STP Root Guard ○

Selected 2 (Total: 13)

□ All
□ 01VLAN VLANID: 150
□ fdg VLANID: 232
□ vlan211 VLANID: 211
☑ vlan222 VLANID: $(vlan222)
□ zd VLANID: 111

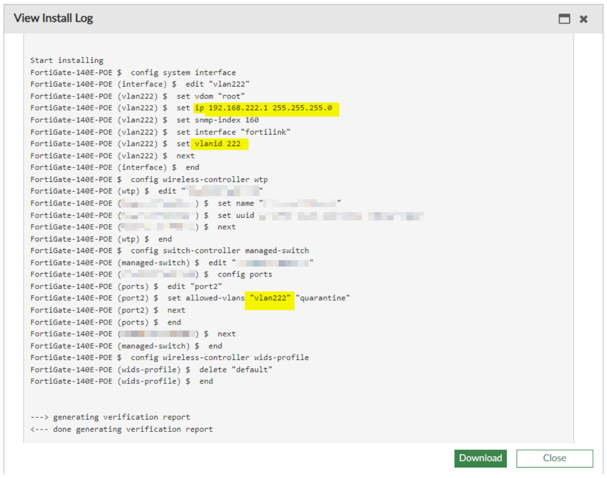


Install Preview of FortiGate-140E-POE

Assigned Devices: FortiGate-140E-POE

FortiGate-140E-POE

```
config system interface
    edit "vlan222"
        set vdom "root"
        set ip 192.168.222.1 255.255.255.0
        set snmp-index 160
        set interface "fortilink"
        set vlanid 222
    next
end
config wireless-controller wtp
    edit "[redacted]"
        set name "[redacted]"
        set uuid [redacted]
    next
end
config switch-controller managed-switch
    edit "[redacted]"
```



```
Start installing
FortiGate-140E-POE $  config system interface
FortiGate-140E-POE (interface) $  edit "vlan222"
FortiGate-140E-POE (vlan222) $  set vdom "root"
FortiGate-140E-POE (vlan222) $  set ip 192.168.222.1 255.255.255.0
FortiGate-140E-POE (vlan222) $  set snmp-index 160
FortiGate-140E-POE (vlan222) $  set interface "fortilink"
FortiGate-140E-POE (vlan222) $  set vlanid 222
FortiGate-140E-POE (vlan222) $  next
FortiGate-140E-POE (interface) $  end
FortiGate-140E-POE $  config wireless-controller wtp
FortiGate-140E-POE (wtp) $  edit "[redacted]"
FortiGate-140E-POE ([redacted]) $  set name "[redacted]"
FortiGate-140E-POE ([redacted]) $  set uuid [redacted]
FortiGate-140E-POE ([redacted]) $  next
FortiGate-140E-POE (wtp) $  end
FortiGate-140E-POE $  config switch-controller managed-switch
FortiGate-140E-POE (managed-switch) $  edit "[redacted]"
FortiGate-140E-POE ([redacted]) $  config ports
FortiGate-140E-POE (ports) $  edit "port2"
FortiGate-140E-POE (port2) $  set allowed-vlans "vlan222" "quarantine"
FortiGate-140E-POE (port2) $  next
FortiGate-140E-POE (ports) $  end
FortiGate-140E-POE ([redacted]) $  next
FortiGate-140E-POE (managed-switch) $  end
FortiGate-140E-POE $  config wireless-controller wids-profile
FortiGate-140E-POE (wids-profile) $  delete "default"
FortiGate-140E-POE (wids-profile) $  end


---> generating verification report
<--- done generating verification report
```

FortiSwitch devices can be imported from a CSV file - FMG 7.4.2

This information is also available in the FortiManager 7.4 Administration Guide:

- Importing and exporting FortiSwitch devices

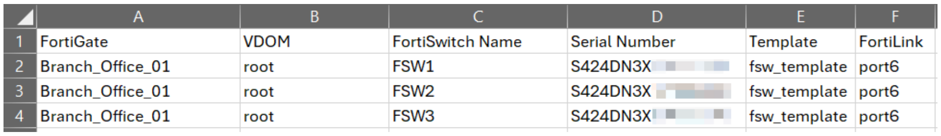
FortiSwitch devices can be imported from a CSV file to provision the switch infrastructure.

To import FortiSwitch devices:

1. Create a CSV file to be imported with the following information:

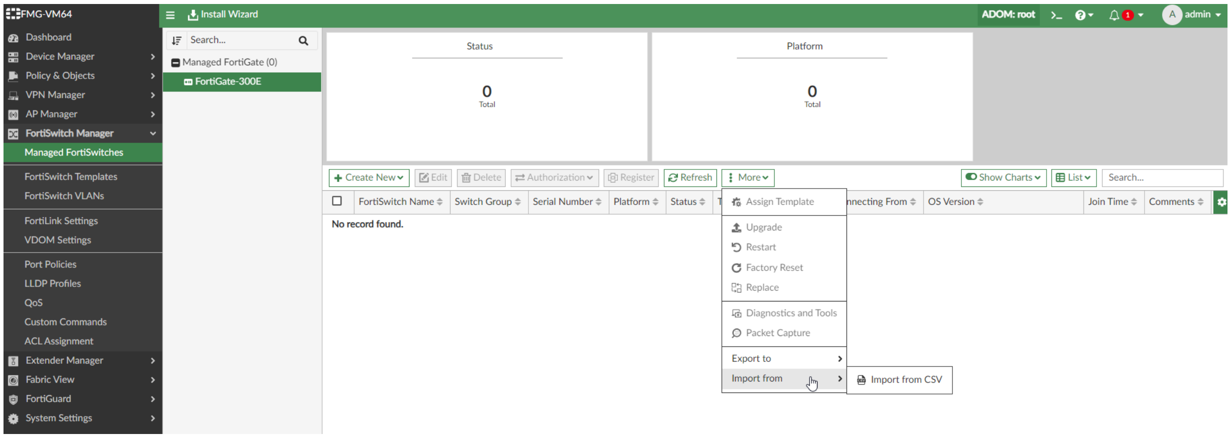
Header	Cell
FortiGate	The name of the FortiGate to which the FortiSwitch will be assigned.
FortiSwitch Name	The FortiSwitches name.
Serial Number	The FortiSwitches serial number
FortiLink	The FortiLink interface used to allow the FortiGate to manage the FortiSwitch.
Template	The template to be assigned to the FortiSwitch.
VDOM	(Optional) If VDOMs are enabled on the FortiGate, specify the VDOM to which the FortiSwitch will be assigned. If VDOMs are disabled, leave this field blank, and the default *root* VDOM will be applied automatically.

For example:



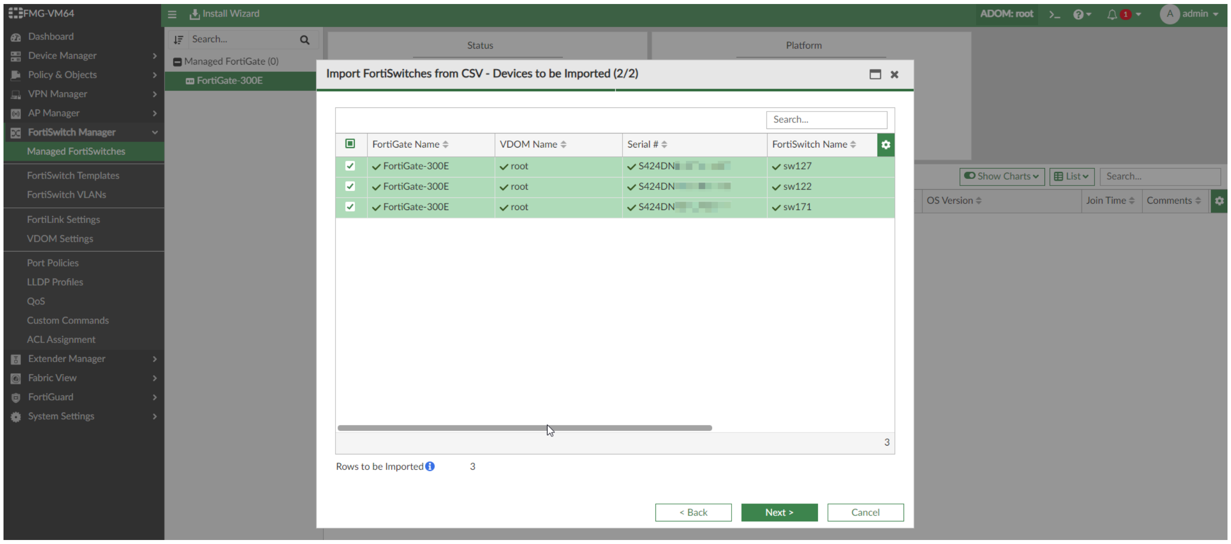
	A	B	C	D	E	F
1	FortiGate	VDOM	FortiSwitch Name	Serial Number	Template	FortiLink
2	Branch_Office_01	root	FSW1	S424DN3X[redacted]	fsw_template	port6
3	Branch_Office_01	root	FSW2	S424DN3X[redacted]	fsw_template	port6
4	Branch_Office_01	root	FSW3	S424DN3X[redacted]	fsw_template	port6

2. Go to *FortiSwitch Manager > Managed FortiSwitches*, and select *More > Import from CSV* from the toolbar.



3. Browse to the CSV file location, or drag and drop the file into the *Upload* field. The results are displayed in the import results window.

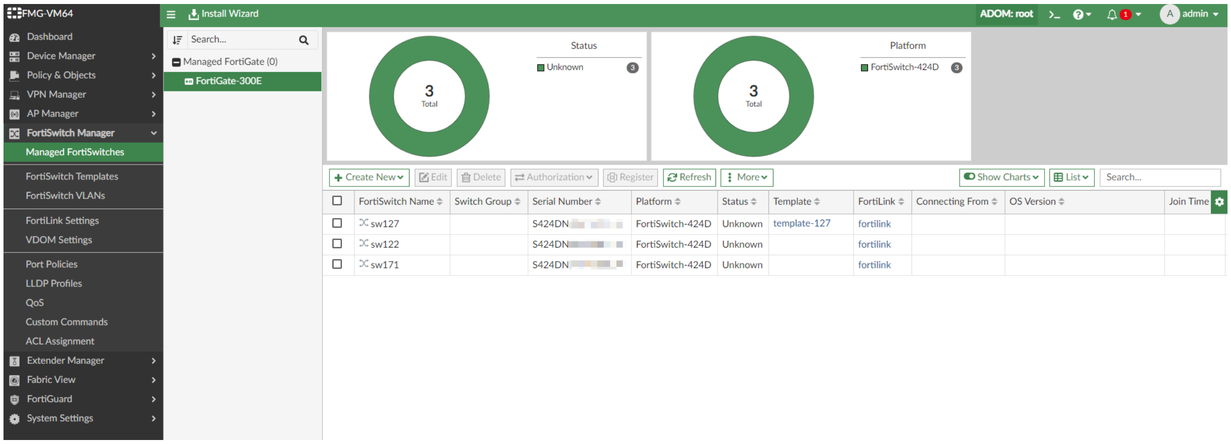
- During import, the validation of the entries will be checked. Successfully imported fields are indicated with a checkmark icon.


Import FortiSwitches from CSV - Devices to be Imported (2/2)

☑	FortiGate Name	VDOM Name	Serial #	FortiSwitch Name
☑	✓ FortiGate-300E	✓ root	✓ S424DN[redacted]	✓ sw127
☑	✓ FortiGate-300E	✓ root	✓ S424DN[redacted]	✓ sw122
☑	✓ FortiGate-300E	✓ root	✓ S424DN[redacted]	✓ sw171

Rows to be Imported 3

- If all entries are valid, the import is successful.



□	FortiSwitch Name	Switch Group	Serial Number	Platform	Status	Template	FortiLink	Connecting From	OS Version	Join Time
□	sw127		S424DN[redacted]	FortiSwitch-424D	Unknown	template-127	fortilink			
□	sw122		S424DN[redacted]	FortiSwitch-424D	Unknown		fortilink			
□	sw171		S424DN[redacted]	FortiSwitch-424D	Unknown		fortilink			

Factory default SSIDs and AP Profiles configuration updated - FMG 7.4.2

This information is also available in the FortiManager 7.4 Administration Guide:

- AP Manager
- FortiSwitch Manager

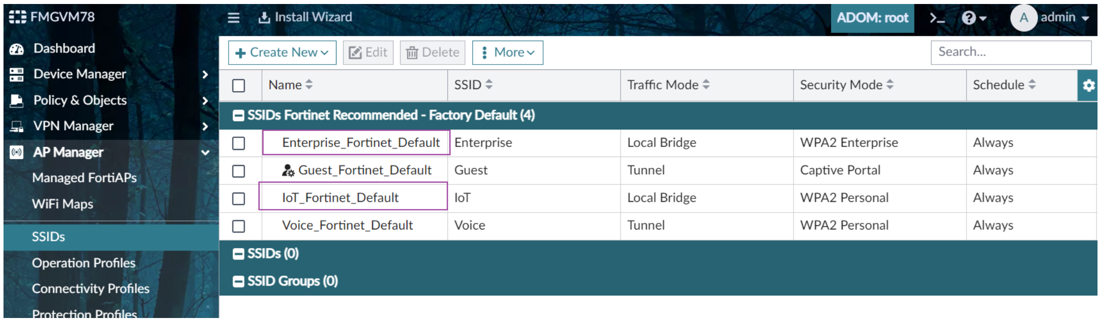
Factory Default SSIDs and AP Profiles configuration updated with optimized settings for performance, increased security and a new Voice SSID.

- SSID templates on page 109
- FortiAP profiles on page 111
- ARRP profile on page 118

SSID templates

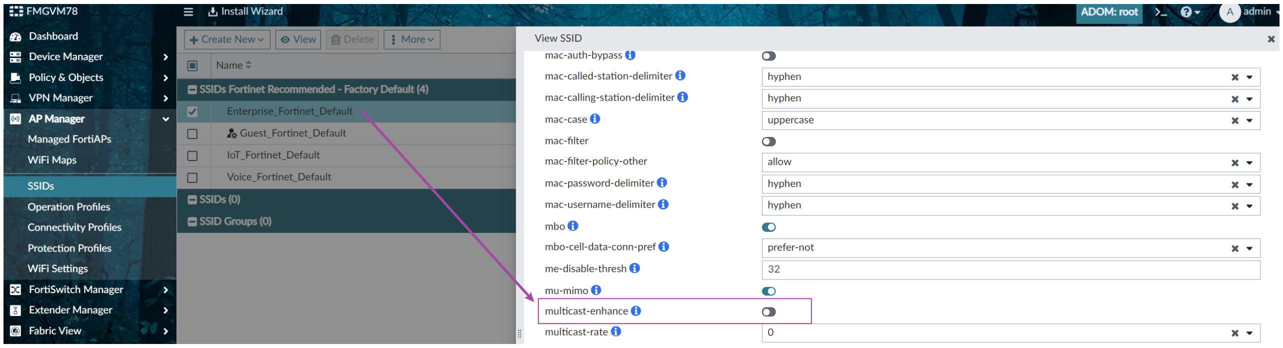
The following enhancements have been made to SSID templates:

1. Existing default SSID templates have been renamed:
   - *Corporate_Fortinet_Default* renamed to *Enterprise_Fortinet_Default*.
   - *POS_Fortinet_Default* renamed to *IoT_Fortinet_Default*.



Name	SSID	Traffic Mode	Security Mode	Schedule
SSIDs Fortinet Recommended - Factory Default (4)				
Enterprise_Fortinet_Default	Enterprise	Local Bridge	WPA2 Enterprise	Always
Guest_Fortinet_Default	Guest	Tunnel	Captive Portal	Always
IoT_Fortinet_Default	IoT	Local Bridge	WPA2 Personal	Always
Voice_Fortinet_Default	Voice	Tunnel	WPA2 Personal	Always
SSIDs (0)				
SSID Groups (0)				

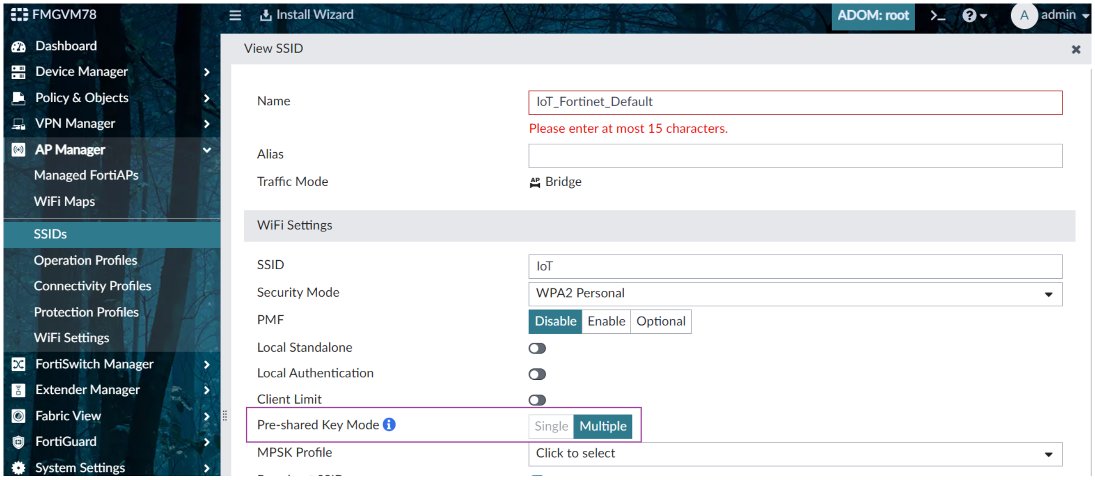
2. For all default SSIDs, *multicast-enhance* is now disabled by default.



View SSID

mac-auth-bypass	○
mac-called-station-delimiter	hyphen
mac-calling-station-delimiter	hyphen
mac-case	uppercase
mac-filter	○
mac-filter-policy-other	allow
mac-password-delimiter	hyphen
mac-username-delimiter	hyphen
mbo	●
mbo-cell-data-conn-pref	prefer-not
me-disable-thresh	32
mu-mimo	●
multicast-enhance	○
multicast-rate	0

3. In the *IoT_Fortinet_Default* SSID, the *Pre-shared Key Mode* setting is set to multiple.



View SSID

Name: IoT_Fortinet_Default

Please enter at most 15 characters.

Alias:

Traffic Mode: Bridge

WiFi Settings

SSID: IoT

Security Mode: WPA2 Personal

PMF: Disable | Enable | Optional

Local Standalone

Local Authentication

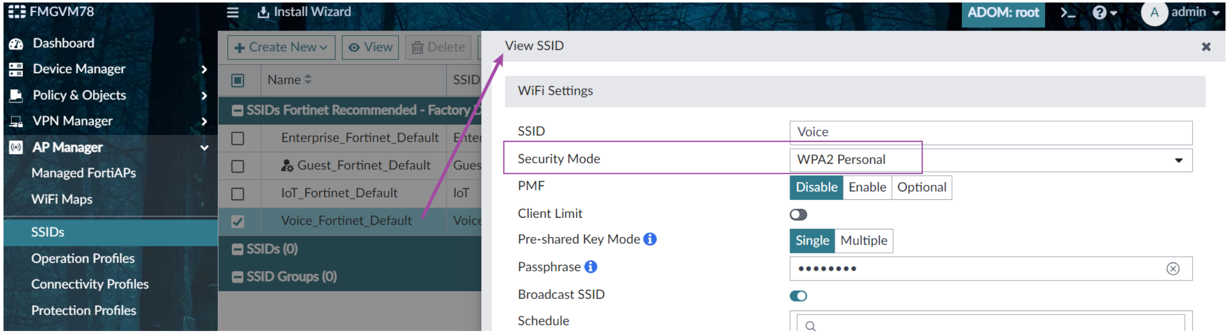
Client Limit

Pre-shared Key Mode: Single | Multiple

MPSK Profile: Click to select

4. Added a new SSID template called *Voice_Fortinet_Default* with the following configuration.

- *Security Mode* set to WPA2-Personal.



View SSID

WiFi Settings

SSID: Voice

Security Mode: WPA2 Personal

PMF: Disable | Enable | Optional

Client Limit

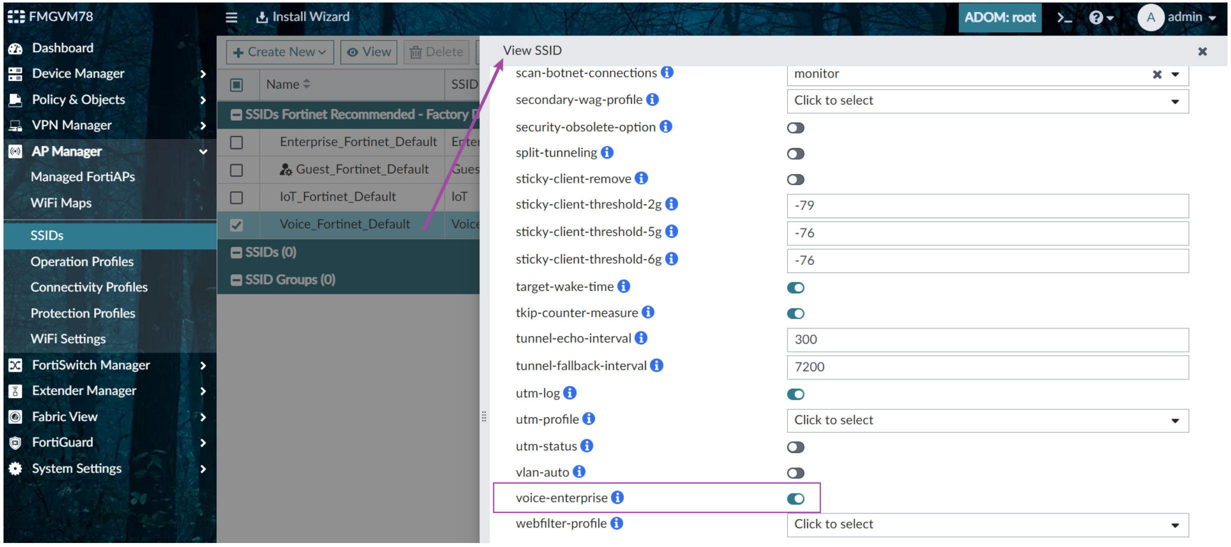
Pre-shared Key Mode: Single | Multiple

Passphrase: ••••••••

Broadcast SSID

Schedule

- Enable *voice-enterprise* (802.11k/v).



View SSID

scan-botnet-connections: monitor

secondary-wag-profile: Click to select

security-obsolete-option

split-tunneling

sticky-client-remove

sticky-client-threshold-2g: -79

sticky-client-threshold-5g: -76

sticky-client-threshold-6g: -76

target-wake-time

tkip-counter-measure

tunnel-echo-interval: 300

tunnel-fallback-interval: 7200

utm-log

utm-profile: Click to select

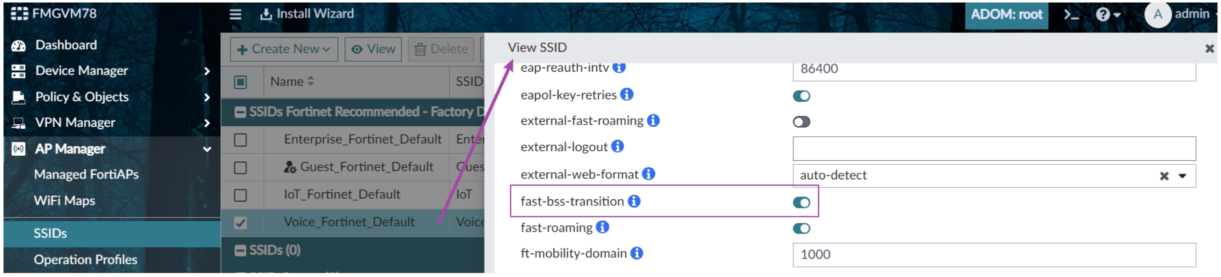
utm-status

vlan-auto

voice-enterprise

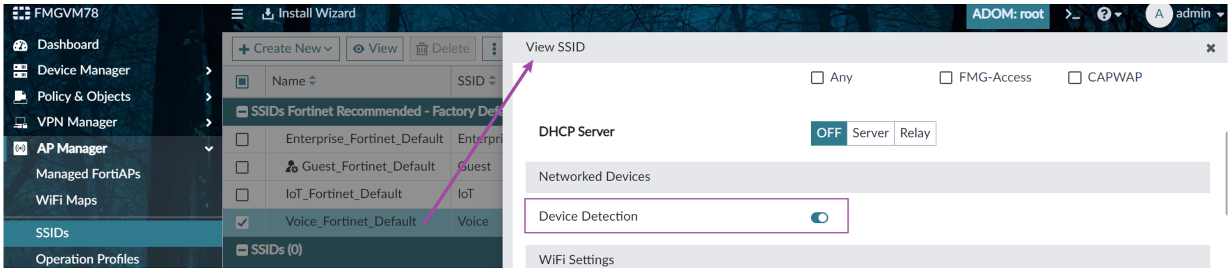
webfilter-profile: Click to select

- Enable *fast-bss-transition* (802.11.r).



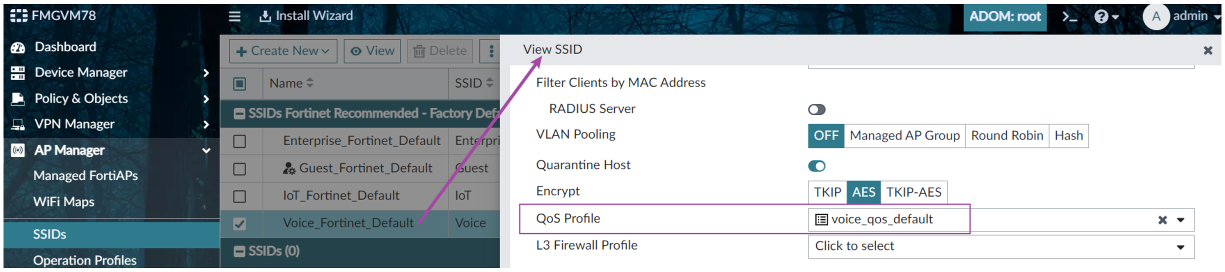
eap-reauth-intv: 86400
eapol-key-retries: ●
external-fast-roaming: ○
external-logout:
external-web-format: auto-detect
fast-bss-transition: ●
fast-roaming: ●
ft-mobility-domain: 1000

- Set *Device Detection* to enable.



☐ Any ☐ FMG-Access ☐ CAPWAP
DHCP Server: [✓ OFF] Server Relay
Networked Devices
Device Detection: ●
WiFi Settings

- Set a default *QoS Profile*.



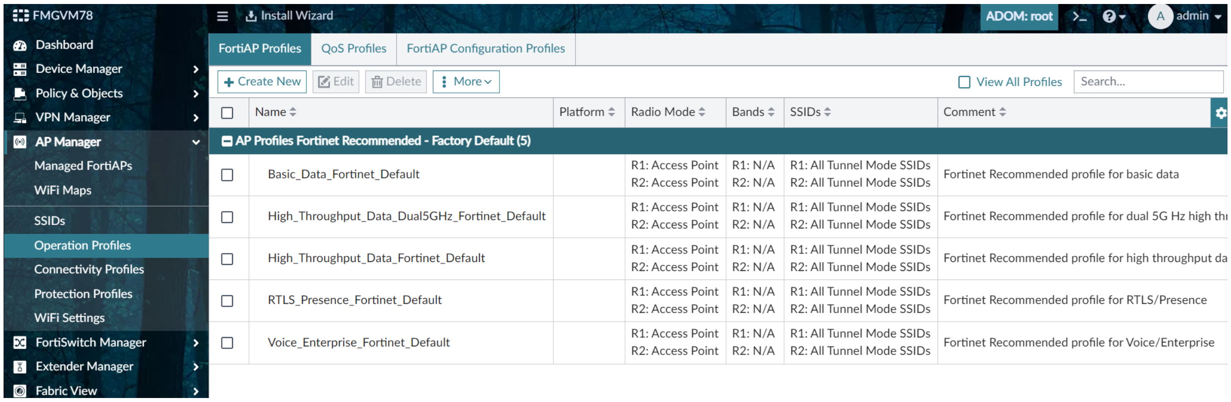
Filter Clients by MAC Address
RADIUS Server: ○
VLAN Pooling: [✓ OFF] Managed AP Group Round Robin Hash
Quarantine Host: ●
Encrypt: TKIP [✓ AES] TKIP-AES
QoS Profile: voice_qos_default
L3 Firewall Profile: Click to select

FortiAP profiles

The following updates have been made for FortiAP Profiles:

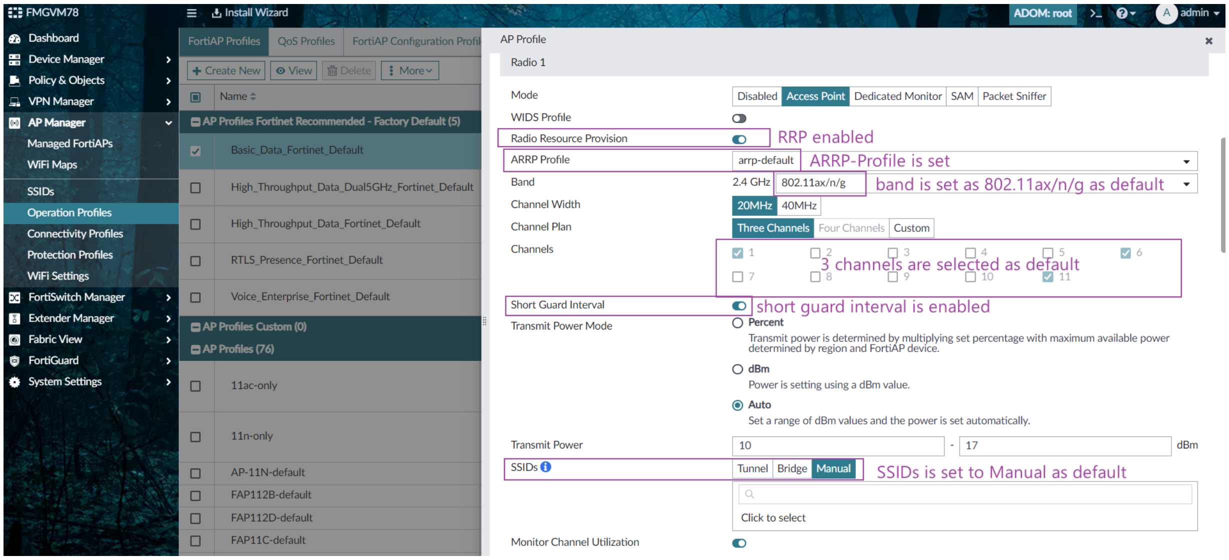
1. The following AP profiles are removed:
   - *Corporate_Fortinet_Default*
   - *Guest_Fortinet_Default*
   - *POS_Fortinet_Default*
2. The following new AP profiles are introduced:
   - Voice/Enterprise_Fortinet_Default
   - Basic_Data_Fortinet_Default
   - High_Throughput_Data_Fortinet_Default
   - High_Throughput_Data_Dual5GHz_Fortinet_Default

- RTLS/Presence_Fortinet_Default



☐	Name	Platform	Radio Mode	Bands	SSIDs	Comment
AP Profiles Fortinet Recommended - Factory Default (5)						
☐	Basic_Data_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for basic data
☐	High_Throughput_Data_Dual5GHz_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for dual 5G Hz high thr
☐	High_Throughput_Data_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for high throughput da
☐	RTLS_Presence_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for RTLS/Presence
☐	Voice_Enterprise_Fortinet_Default		R1: Access Point R2: Access Point	R1: N/A R2: N/A	R1: All Tunnel Mode SSIDs R2: All Tunnel Mode SSIDs	Fortinet Recommended profile for Voice/Enterprise

3. The following configurations are common to all default AP Profiles:
   - Set 2.4GHz radio band to *802.11ax/g/n* (exclude b).
   - Set 2.4GHz channel width *20MHz*.
   - Set 2.4GHz channel plan to *Three Channels* by default.
   - Set ARRP-Profile to *arrp-default*.
   - Set Short Guard Interval to enable on all radios.
   - Set Radio Resource Provisioning to enable on all radios.
   - Set SSIDs to *Manual* for all radios.



AP Profile

Radio 1

Mode: Disabled | [✓ Access Point] | Dedicated Monitor | SAM | Packet Sniffer

WIDS Profile

Radio Resource Provision — RRP enabled

ARRP Profile: arrp-default — ARRP-Profile is set

Band: 2.4 GHz 802.11ax/n/g — band is set as 802.11ax/n/g as default

Channel Width: [✓ 20MHz] | 40MHz

Channel Plan: [✓ Three Channels] | Four Channels | Custom

Channels: ☑ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☑ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10 ☑ 11 — 3 channels are selected as default

Short Guard Interval — short guard interval is enabled

Transmit Power Mode:
○ Percent
Transmit power is determined by multiplying set percentage with maximum available power determined by region and FortiAP device.
○ dBm
Power is setting using a dBm value.
● Auto
Set a range of dBm values and the power is set automatically.

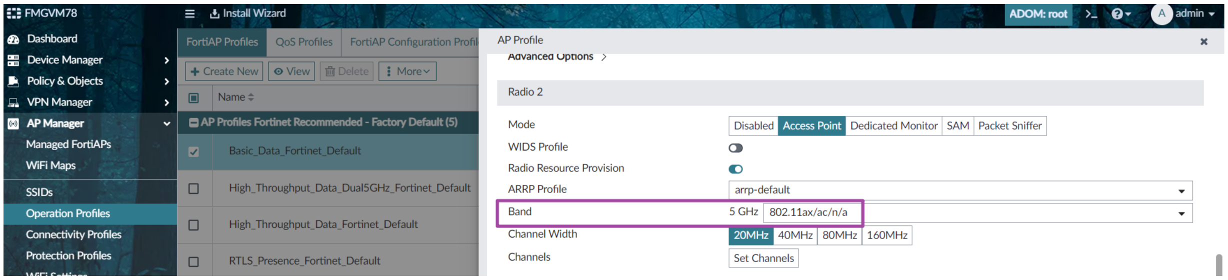
Transmit Power: 10 - 17 dBm

SSIDs: Tunnel | Bridge | [✓ Manual] — SSIDs is set to Manual as default

Click to select

Monitor Channel Utilization

- 5GHz set to default *802.11ax/ac/n/a* - include a (matching g data rates).



AP Profile

Advanced Options

Radio 2

Mode: Disabled | [✓ Access Point] | Dedicated Monitor | SAM | Packet Sniffer

WIDS Profile

Radio Resource Provision

ARRP Profile: arrp-default

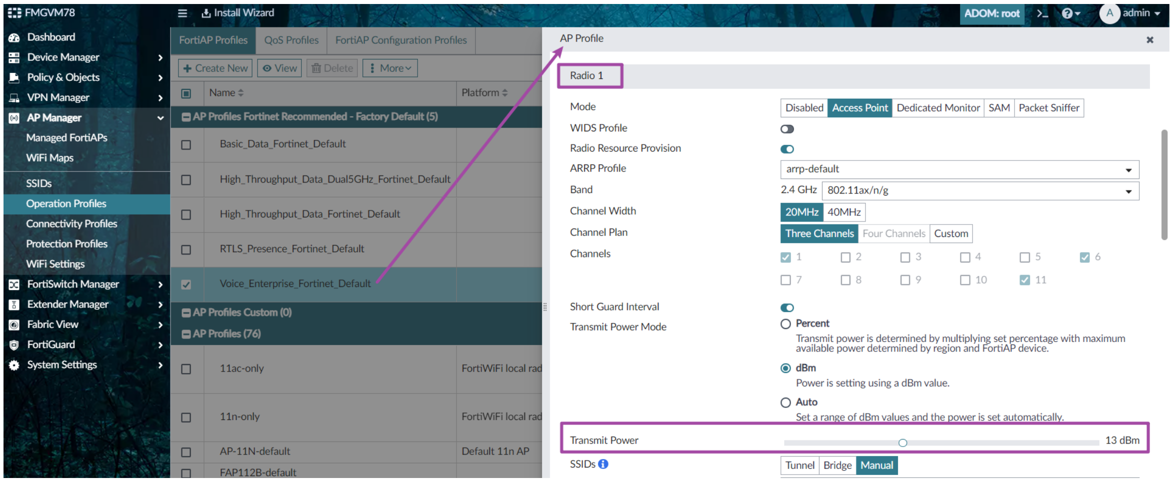
Band: 5 GHz 802.11ax/ac/n/a

Channel Width: [✓ 20MHz] | 40MHz | 80MHz | 160MHz

Channels: Set Channels

4. Specific configurations are also applied to each AP profile:

- **Voice/Enterprise_Fortinet_Default:**
  - Set channel width to 20MHz for 5GHz band
  - Set channel width to 40MHz on 6GHz (if applicable AP model selected)
  - 2.4GHz Transmit Power Mode - dBm - 13dBm
  - 5GHz Transmit Power Mode - dBm - 16dBm
  - 6GHz Transmit Power Mode - dBm – 19dBm



AP Profiles Fortinet Recommended - Factory Default (5)
- □ Basic_Data_Fortinet_Default
- □ High_Throughput_Data_Dual5GHz_Fortinet_Default
- □ High_Throughput_Data_Fortinet_Default
- □ RTLS_Presence_Fortinet_Default
- ☑ Voice_Enterprise_Fortinet_Default

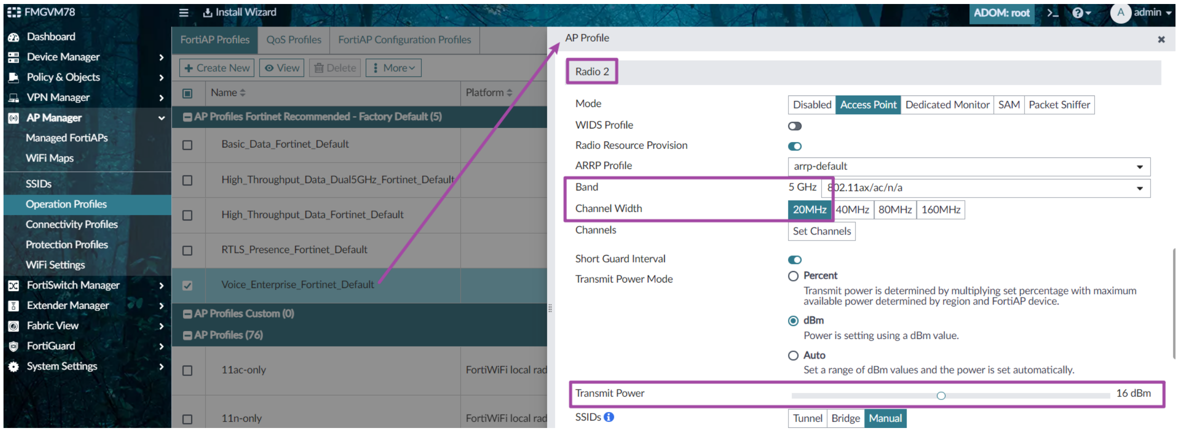
AP Profiles Custom (0)

AP Profiles (76)
- □ 11ac-only — FortiWiFi local rad
- □ 11n-only — FortiWiFi local rad
- □ AP-11N-default — Default 11n AP
- □ FAP112B-default

AP Profile

Radio 1

Mode: Disabled | [✓ Access Point] | Dedicated Monitor | SAM | Packet Sniffer
WIDS Profile
Radio Resource Provision
ARRP Profile: arrp-default
Band: 2.4 GHz 802.11ax/n/g
Channel Width: [✓ 20MHz] 40MHz
Channel Plan: [✓ Three Channels] Four Channels Custom
Channels: ☑ 1 □ 2 □ 3 □ 4 □ 5 ☑ 6 □ 7 □ 8 □ 9 □ 10 ☑ 11
Short Guard Interval
Transmit Power Mode:
○ Percent
Transmit power is determined by multiplying set percentage with maximum available power determined by region and FortiAP device.
● dBm
Power is setting using a dBm value.
○ Auto
Set a range of dBm values and the power is set automatically.
Transmit Power: 13 dBm
SSIDs: Tunnel | Bridge | [✓ Manual]



AP Profiles Fortinet Recommended - Factory Default (5)
- □ Basic_Data_Fortinet_Default
- □ High_Throughput_Data_Dual5GHz_Fortinet_Default
- □ High_Throughput_Data_Fortinet_Default
- □ RTLS_Presence_Fortinet_Default
- ☑ Voice_Enterprise_Fortinet_Default

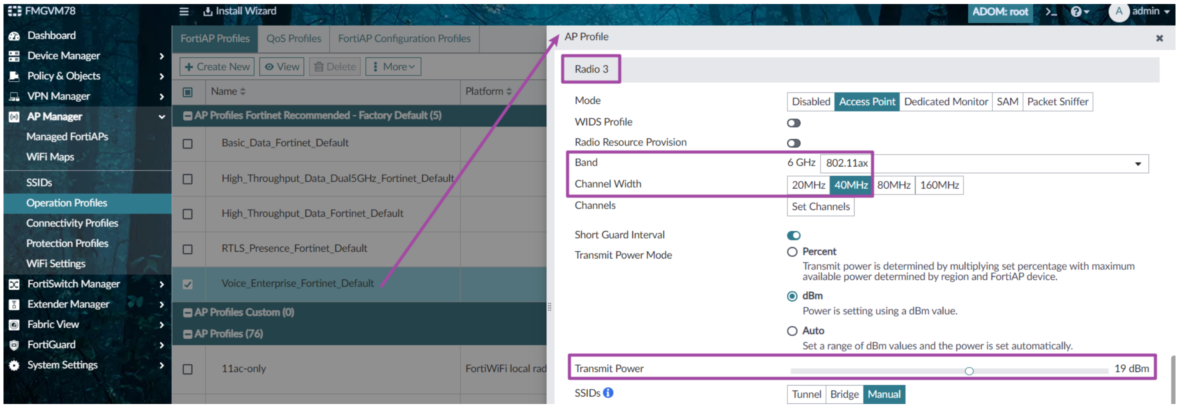
AP Profiles Custom (0)

AP Profiles (76)
- □ 11ac-only — FortiWiFi local rad
- □ 11n-only — FortiWiFi local rad

AP Profile

Radio 2

Mode: Disabled | [✓ Access Point] | Dedicated Monitor | SAM | Packet Sniffer
WIDS Profile
Radio Resource Provision
ARRP Profile: arrp-default
Band: 5 GHz 802.11ax/ac/n/a
Channel Width: [✓ 20MHz] 40MHz 80MHz 160MHz
Channels: Set Channels
Short Guard Interval
Transmit Power Mode:
○ Percent
Transmit power is determined by multiplying set percentage with maximum available power determined by region and FortiAP device.
● dBm
Power is setting using a dBm value.
○ Auto
Set a range of dBm values and the power is set automatically.
Transmit Power: 16 dBm
SSIDs: Tunnel | Bridge | [✓ Manual]



AP Profiles Fortinet Recommended - Factory Default (5)
- □ Basic_Data_Fortinet_Default
- □ High_Throughput_Data_Dual5GHz_Fortinet_Default
- □ High_Throughput_Data_Fortinet_Default
- □ RTLS_Presence_Fortinet_Default
- ☑ Voice_Enterprise_Fortinet_Default

AP Profiles Custom (0)

AP Profiles (76)
- □ 11ac-only — FortiWiFi local rad

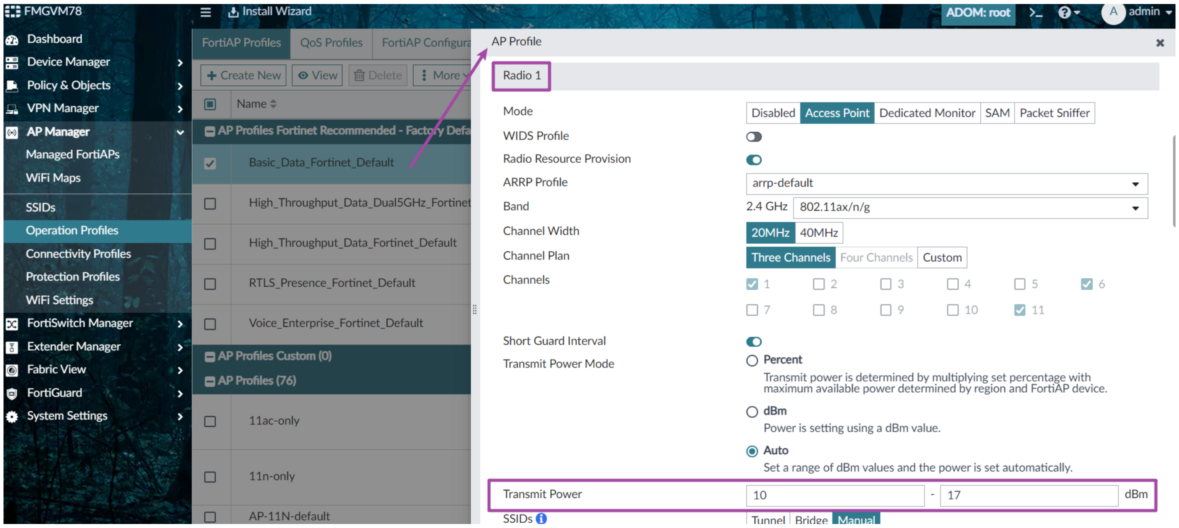
AP Profile

Radio 3

Mode: Disabled | [✓ Access Point] | Dedicated Monitor | SAM | Packet Sniffer
WIDS Profile
Radio Resource Provision
Band: 6 GHz 802.11ax
Channel Width: 20MHz [✓ 40MHz] 80MHz 160MHz
Channels: Set Channels
Short Guard Interval
Transmit Power Mode:
○ Percent
Transmit power is determined by multiplying set percentage with maximum available power determined by region and FortiAP device.
● dBm
Power is setting using a dBm value.
○ Auto
Set a range of dBm values and the power is set automatically.
Transmit Power: 19 dBm
SSIDs: Tunnel | Bridge | [✓ Manual]

- **Basic_Data_Fortinet_Default**
  - Set channel width to 20MHz for 5GHz band
  - Set channel width to 40MHz on 6GHz (if applicable AP model selected)

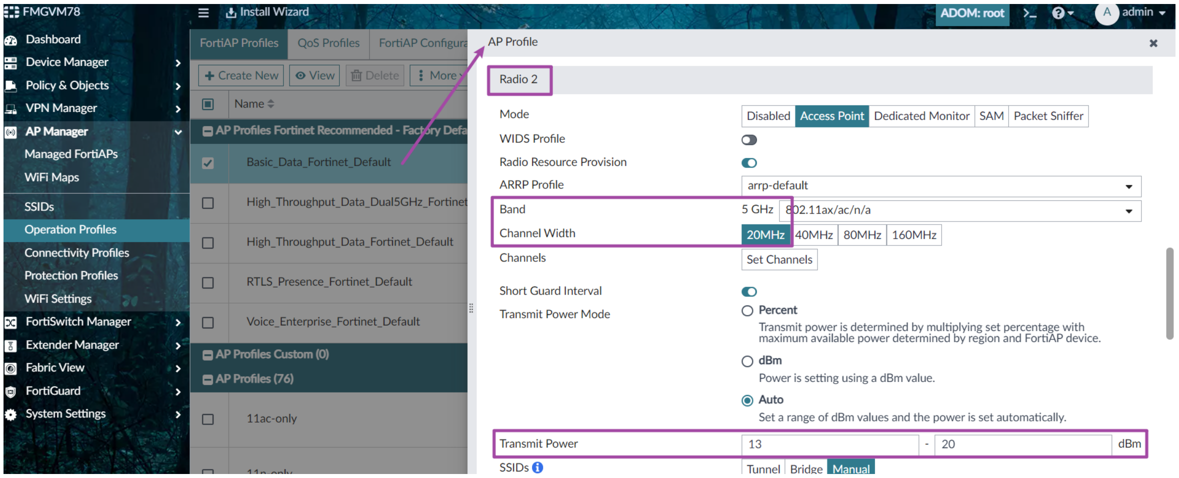
- 2.4GHz Transmit Power Mode - Auto – 10 to 17 dBm
- 5GHz Transmit Power Mode - Auto - 13 to 20 dBm
- 6GHz Transmit Power Mode - Auto - 16 to 23 dBm



AP Profile

Radio 1

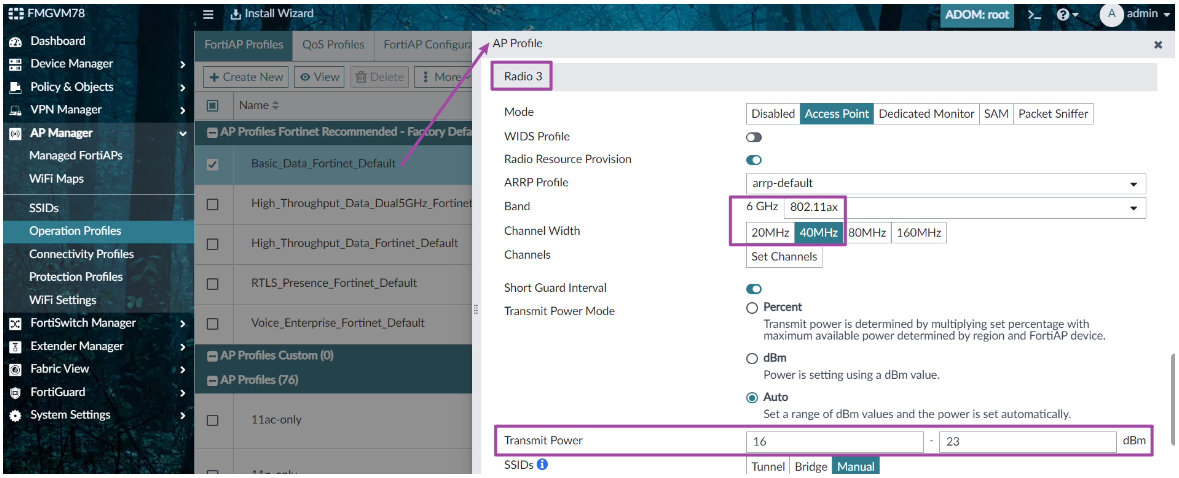
Mode: Disabled | [✓ Access Point] | Dedicated Monitor | SAM | Packet Sniffer
WIDS Profile
Radio Resource Provision
ARRP Profile: arrp-default
Band: 2.4 GHz 802.11ax/n/g
Channel Width: [✓ 20MHz] 40MHz
Channel Plan: [✓ Three Channels] Four Channels Custom
Channels: ☑ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☑ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10 ☑ 11
Short Guard Interval
Transmit Power Mode:
○ Percent
Transmit power is determined by multiplying set percentage with maximum available power determined by region and FortiAP device.
○ dBm
Power is setting using a dBm value.
● Auto
Set a range of dBm values and the power is set automatically.
Transmit Power: 10 - 17 dBm
SSIDs: Tunnel Bridge [✓ Manual]



AP Profile

Radio 2

Mode: Disabled | [✓ Access Point] | Dedicated Monitor | SAM | Packet Sniffer
WIDS Profile
Radio Resource Provision
ARRP Profile: arrp-default
Band: 5 GHz 802.11ax/ac/n/a
Channel Width: [✓ 20MHz] 40MHz 80MHz 160MHz
Channels: Set Channels
Short Guard Interval
Transmit Power Mode:
○ Percent
Transmit power is determined by multiplying set percentage with maximum available power determined by region and FortiAP device.
○ dBm
Power is setting using a dBm value.
● Auto
Set a range of dBm values and the power is set automatically.
Transmit Power: 13 - 20 dBm
SSIDs: Tunnel Bridge [✓ Manual]



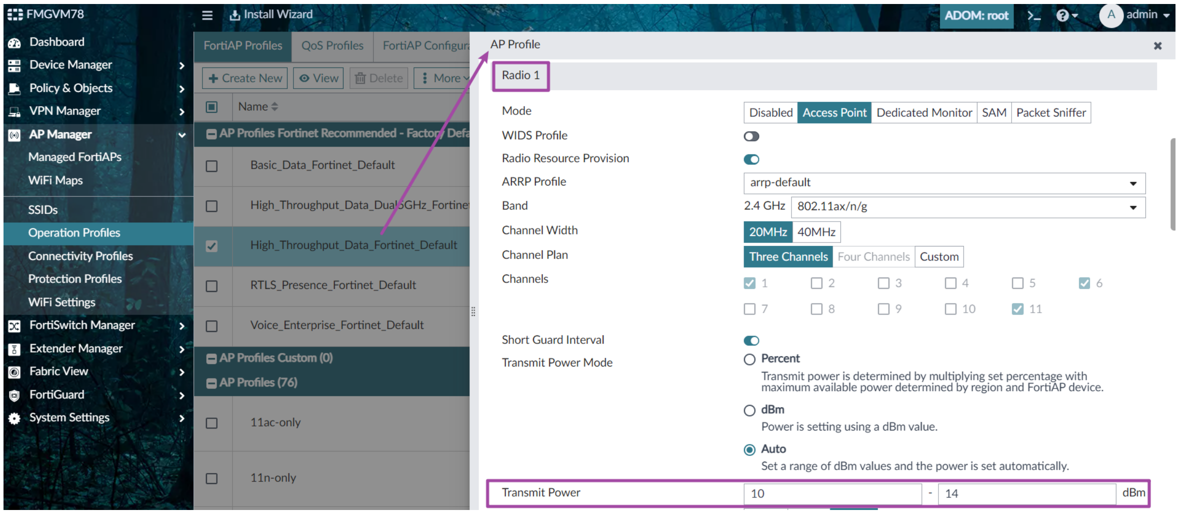
AP Profile

Radio 3

Mode: Disabled | [✓ Access Point] | Dedicated Monitor | SAM | Packet Sniffer
WIDS Profile
Radio Resource Provision
ARRP Profile: arrp-default
Band: 6 GHz 802.11ax
Channel Width: 20MHz [✓ 40MHz] 80MHz 160MHz
Channels: Set Channels
Short Guard Interval
Transmit Power Mode:
○ Percent
Transmit power is determined by multiplying set percentage with maximum available power determined by region and FortiAP device.
○ dBm
Power is setting using a dBm value.
● Auto
Set a range of dBm values and the power is set automatically.
Transmit Power: 16 - 23 dBm
SSIDs: Tunnel Bridge [✓ Manual]

- **High_Throughput_Data_Fortinet_Default**

- Set 5GHz channel width to 40MHz – Up to 12 available channels including DFS frequencies
- Set 6GHz channel width to 80MHz (if applicable AP model selected) – Up to 14 available channels
- 2.4GHz Transmit Power Mode – Auto – 10dBm to 14dBm
- 5GHz Transmit Power Mode - Auto – 13 to 17dBm
- 6GHz Transmit Power Mode - Auto – 16 to 20dBm (if applicable AP model selected)



Mode: Disabled | **Access Point** | Dedicated Monitor | SAM | Packet Sniffer

WIDS Profile

Radio Resource Provision

ARRP Profile: arrp-default

Band: 2.4 GHz 802.11ax/n/g

Channel Width: **20MHz** | 40MHz

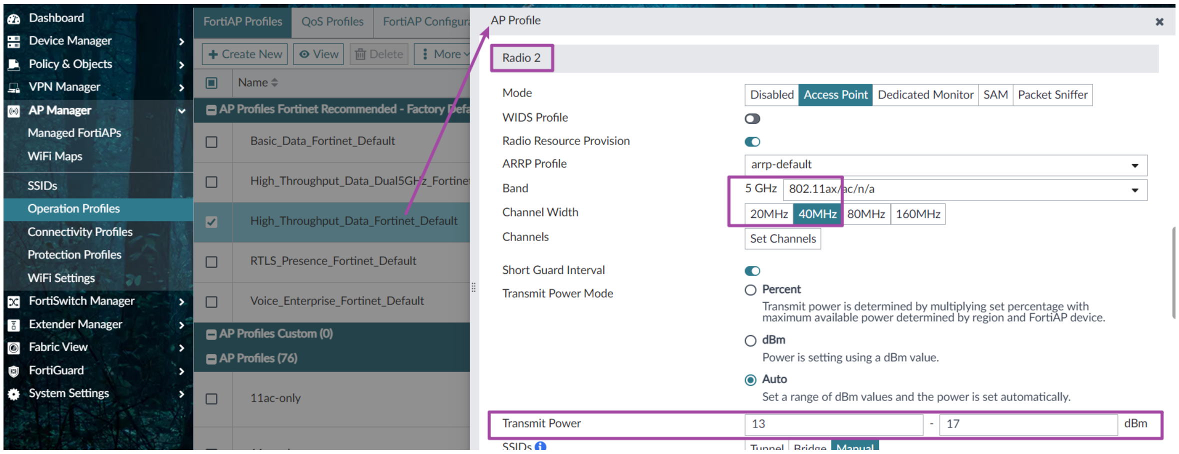
Channel Plan: **Three Channels** | Four Channels | Custom

Channels: ☑ 1 □ 2 □ 3 □ 4 □ 5 ☑ 6 □ 7 □ 8 □ 9 □ 10 ☑ 11

Short Guard Interval

Transmit Power Mode:
○ Percent
Transmit power is determined by multiplying set percentage with maximum available power determined by region and FortiAP device.
○ dBm
Power is setting using a dBm value.
● Auto
Set a range of dBm values and the power is set automatically.

Transmit Power: 10 - 14 dBm



Mode: Disabled | **Access Point** | Dedicated Monitor | SAM | Packet Sniffer

WIDS Profile

Radio Resource Provision

ARRP Profile: arrp-default

Band: 5 GHz 802.11ax/ac/n/a

Channel Width: 20MHz | **40MHz** | 80MHz | 160MHz

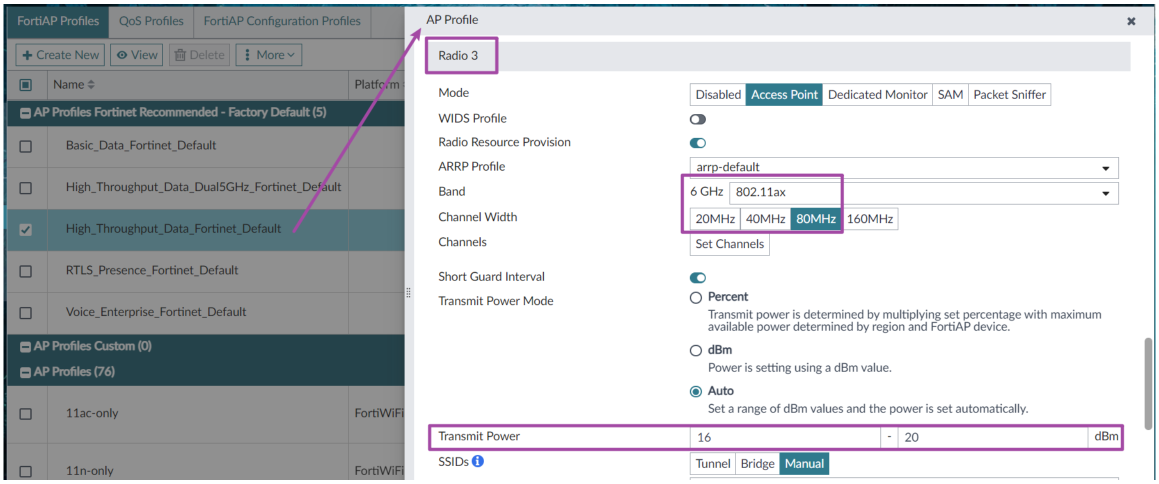
Channels: Set Channels

Short Guard Interval

Transmit Power Mode:
○ Percent
Transmit power is determined by multiplying set percentage with maximum available power determined by region and FortiAP device.
○ dBm
Power is setting using a dBm value.
● Auto
Set a range of dBm values and the power is set automatically.

Transmit Power: 13 - 17 dBm

SSIDs: Tunnel | Bridge | **Manual**


FortiAP Profiles
QoS Profiles
FortiAP Configuration Profiles
Create New
View
Delete
More
Name
Platform
AP Profiles Fortinet Recommended - Factory Default (5)
Basic_Data_Fortinet_Default
High_Throughput_Data_Dual5GHz_Fortinet_Default
High_Throughput_Data_Fortinet_Default
RTLS_Presence_Fortinet_Default
Voice_Enterprise_Fortinet_Default
AP Profiles Custom (0)
AP Profiles (76)
11ac-only
FortiWiFi
11n-only
FortiWiFi
AP Profile
Radio 3

Mode: Disabled | **Access Point** | Dedicated Monitor | SAM | Packet Sniffer

WIDS Profile

Radio Resource Provision

ARRP Profile: arrp-default

Band: 6 GHz 802.11ax

Channel Width: 20MHz | 40MHz | **80MHz** | 160MHz

Channels: Set Channels

Short Guard Interval

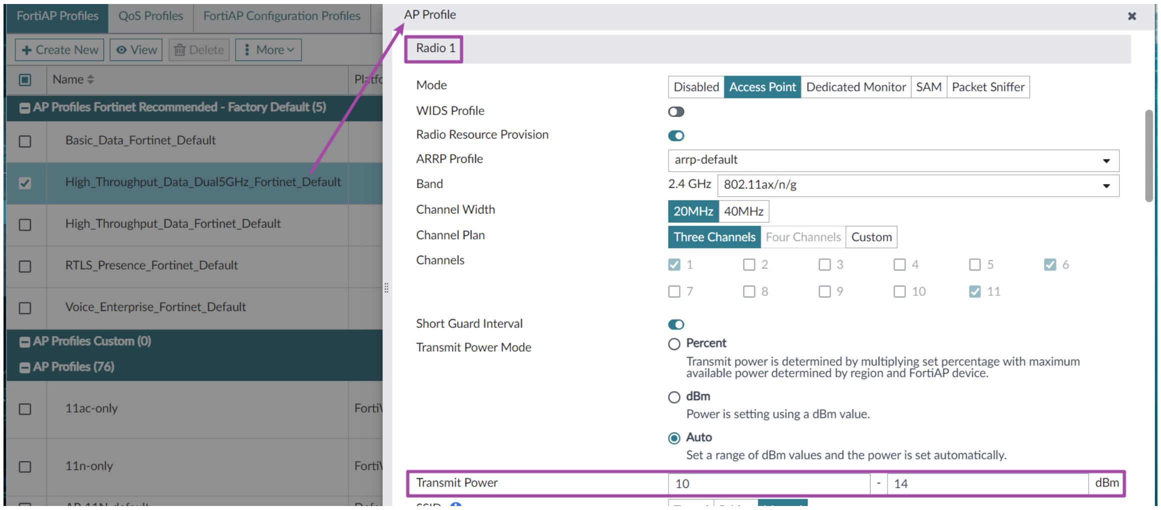
Transmit Power Mode:
○ Percent
Transmit power is determined by multiplying set percentage with maximum available power determined by region and FortiAP device.
○ dBm
Power is setting using a dBm value.
● Auto
Set a range of dBm values and the power is set automatically.

Transmit Power: 16 - 20 dBm

SSIDs: Tunnel | Bridge | **Manual**

- **High_Throughput_Data_Dual5GHz_Fortinet_Default**

- Set 5GHz channel width to 40MHz – Up to 12 available channels including DFS frequencies
- Set 6GHz channel width to 80MHz (if applicable AP model selected) – Up to 14 available channels
- 2.4GHz Transmit Power Mode – Auto – 10dBm to 14dBm
- 5GHz Transmit Power Mode - Auto – 13 to 17dBm
- 5GHz Transmit Power Mode - Auto – 13 to 17dBm (if applicable dual 5GHz AP model selected)



FortiAP Profiles | QoS Profiles | FortiAP Configuration Profiles

+ Create New | View | Delete | More

□ Name | Platfo

AP Profiles Fortinet Recommended - Factory Default (5)

- □ Basic_Data_Fortinet_Default
- ☑ High_Throughput_Data_Dual5GHz_Fortinet_Default
- □ High_Throughput_Data_Fortinet_Default
- □ RTLS_Presence_Fortinet_Default
- □ Voice_Enterprise_Fortinet_Default

AP Profiles Custom (0)

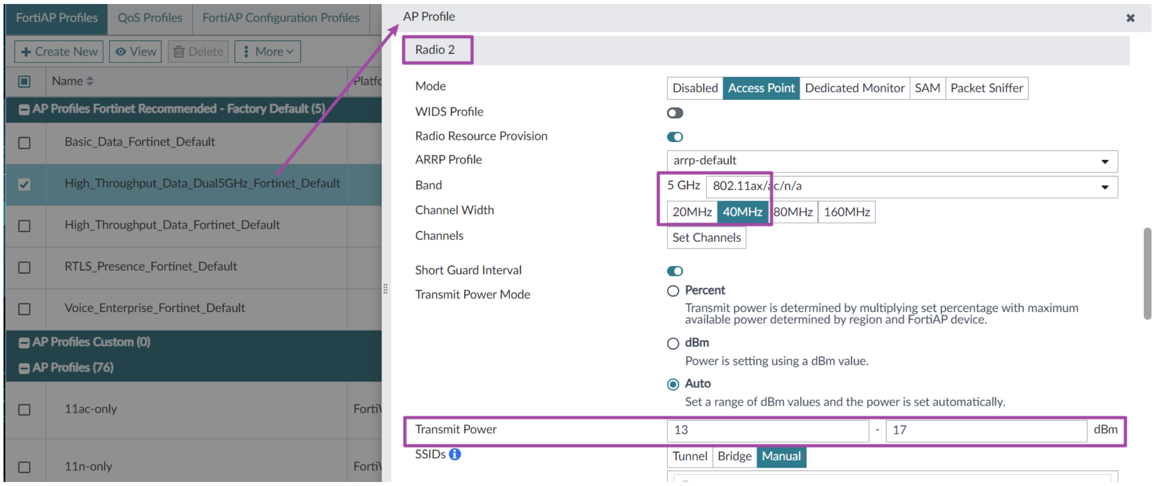
AP Profiles (76)

- □ 11ac-only Forti
- □ 11n-only Forti

AP Profile

Radio 1

Mode: Disabled | **Access Point** | Dedicated Monitor | SAM | Packet Sniffer
WIDS Profile
Radio Resource Provision
ARRP Profile: arrp-default
Band: 2.4 GHz 802.11ax/n/g
Channel Width: **20MHz** | 40MHz
Channel Plan: **Three Channels** | Four Channels | Custom
Channels: ☑ 1 □ 2 □ 3 □ 4 □ 5 ☑ 6 □ 7 □ 8 □ 9 □ 10 ☑ 11
Short Guard Interval
Transmit Power Mode:
○ Percent
Transmit power is determined by multiplying set percentage with maximum available power determined by region and FortiAP device.
○ dBm
Power is setting using a dBm value.
● Auto
Set a range of dBm values and the power is set automatically.
Transmit Power: 10 - 14 dBm



FortiAP Profiles | QoS Profiles | FortiAP Configuration Profiles

+ Create New | View | Delete | More

□ Name | Platfo

AP Profiles Fortinet Recommended - Factory Default (5)

- □ Basic_Data_Fortinet_Default
- ☑ High_Throughput_Data_Dual5GHz_Fortinet_Default
- □ High_Throughput_Data_Fortinet_Default
- □ RTLS_Presence_Fortinet_Default
- □ Voice_Enterprise_Fortinet_Default

AP Profiles Custom (0)

AP Profiles (76)

- □ 11ac-only Forti
- □ 11n-only Forti

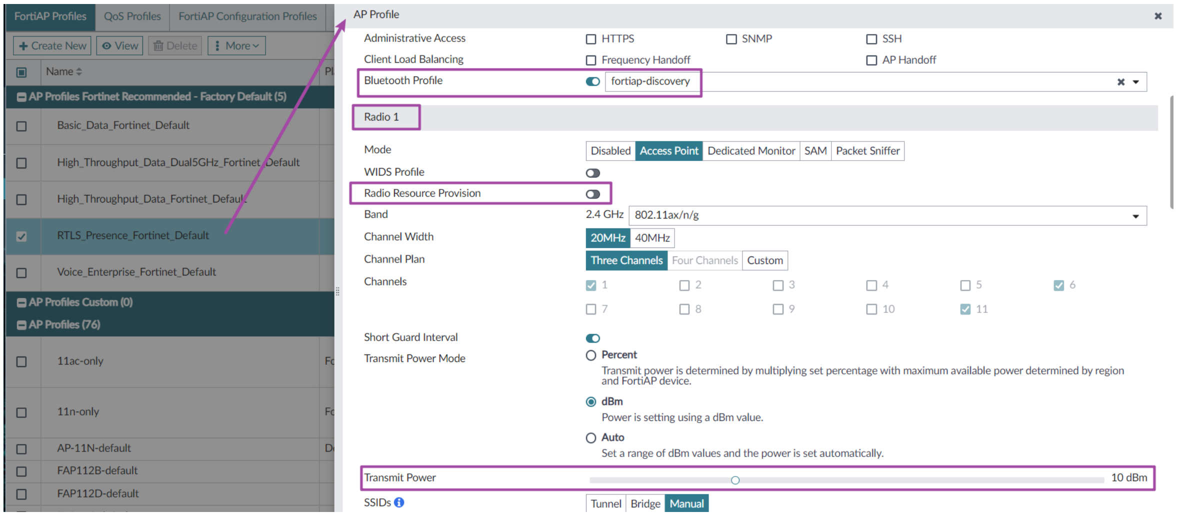
AP Profile

Radio 2

Mode: Disabled | **Access Point** | Dedicated Monitor | SAM | Packet Sniffer
WIDS Profile
Radio Resource Provision
ARRP Profile: arrp-default
Band: 5 GHz 802.11ax/ac/n/a
Channel Width: 20MHz | **40MHz** | 80MHz | 160MHz
Channels: Set Channels
Short Guard Interval
Transmit Power Mode:
○ Percent
Transmit power is determined by multiplying set percentage with maximum available power determined by region and FortiAP device.
○ dBm
Power is setting using a dBm value.
● Auto
Set a range of dBm values and the power is set automatically.
Transmit Power: 13 - 17 dBm
SSIDs: Tunnel | Bridge | **Manual**

- **RTLS/Presence_Fortinet_Default**
  - Set channel width to 20MHz for 5GHz band
  - Set channel width to 40MHz on 6GHz (if applicable AP model selected)
  - 2.4GHz Transmit Power Mode - dBm - 10dBm
  - 5GHz Transmit Power Mode - dBm - 14dBm
  - 6GHz Transmit Power Mode - dBm – 17dBm
  - Location Based Services - Set Locate WiFi Clients When Not Connected to enable
  - BLE Radio enabled

- Disable RRP



FortiAP Profiles | QoS Profiles | FortiAP Configuration Profiles

+ Create New | View | Delete | More

Name

AP Profiles Fortinet Recommended - Factory Default (5)

□ Basic_Data_Fortinet_Default
□ High_Throughput_Data_Dual5GHz_Fortinet_Default
□ High_Throughput_Data_Fortinet_Default
☑ RTLS_Presence_Fortinet_Default
□ Voice_Enterprise_Fortinet_Default

AP Profiles Custom (0)

AP Profiles (76)

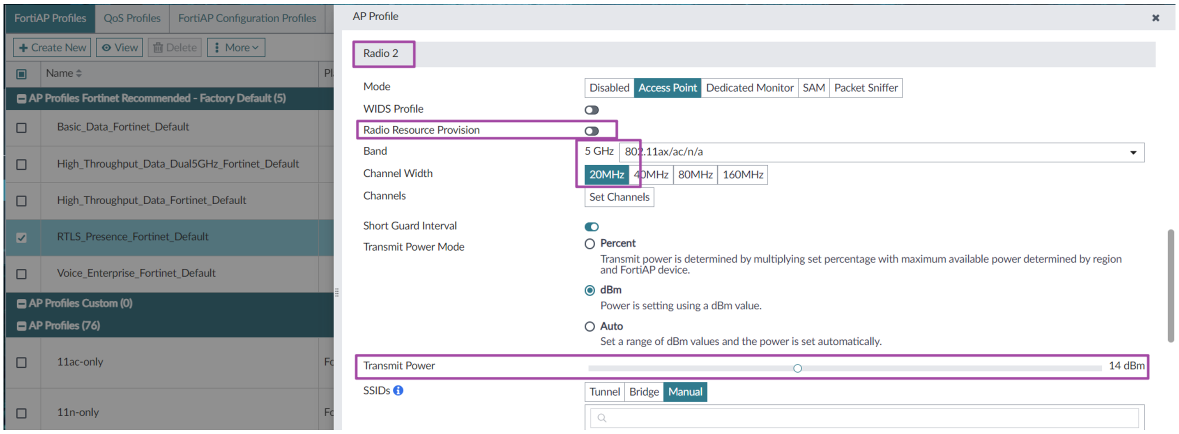
□ 11ac-only
□ 11n-only
□ AP-11N-default
□ FAP112B-default
□ FAP112D-default

AP Profile

Administrative Access: □ HTTPS □ SNMP □ SSH
Client Load Balancing: □ Frequency Handoff □ AP Handoff
Bluetooth Profile: fortiap-discovery

Radio 1

Mode: Disabled | [✓ Access Point] | Dedicated Monitor | SAM | Packet Sniffer
WIDS Profile: off
Radio Resource Provision: off
Band: 2.4 GHz 802.11ax/n/g
Channel Width: [✓ 20MHz] | 40MHz
Channel Plan: [✓ Three Channels] | Four Channels | Custom
Channels: ☑ 1 □ 2 □ 3 □ 4 □ 5 ☑ 6 □ 7 □ 8 □ 9 □ 10 ☑ 11
Short Guard Interval: on
Transmit Power Mode:
○ Percent
Transmit power is determined by multiplying set percentage with maximum available power determined by region and FortiAP device.
● dBm
Power is setting using a dBm value.
○ Auto
Set a range of dBm values and the power is set automatically.
Transmit Power: 10 dBm
SSIDs: Tunnel | Bridge | [✓ Manual]



FortiAP Profiles | QoS Profiles | FortiAP Configuration Profiles

+ Create New | View | Delete | More

Name

AP Profiles Fortinet Recommended - Factory Default (5)

□ Basic_Data_Fortinet_Default
□ High_Throughput_Data_Dual5GHz_Fortinet_Default
□ High_Throughput_Data_Fortinet_Default
☑ RTLS_Presence_Fortinet_Default
□ Voice_Enterprise_Fortinet_Default

AP Profiles Custom (0)

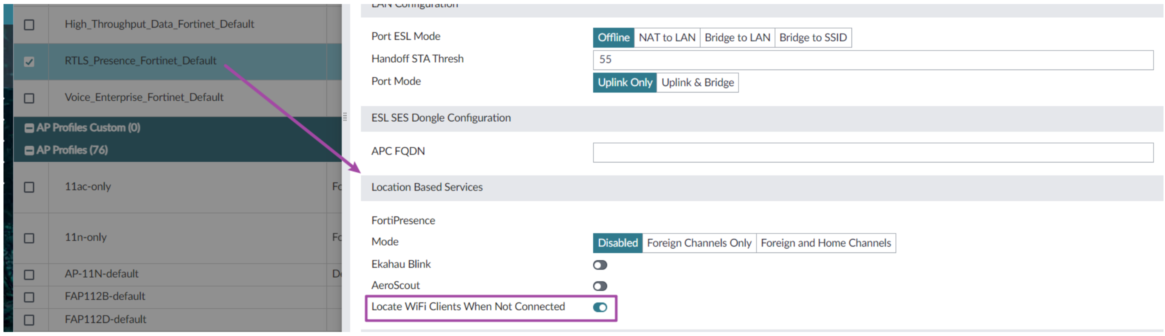
AP Profiles (76)

□ 11ac-only
□ 11n-only

AP Profile

Radio 2

Mode: Disabled | [✓ Access Point] | Dedicated Monitor | SAM | Packet Sniffer
WIDS Profile: off
Radio Resource Provision: off
Band: 5 GHz 802.11ax/ac/n/a
Channel Width: [✓ 20MHz] | 40MHz | 80MHz | 160MHz
Channels: Set Channels
Short Guard Interval: on
Transmit Power Mode:
○ Percent
Transmit power is determined by multiplying set percentage with maximum available power determined by region and FortiAP device.
● dBm
Power is setting using a dBm value.
○ Auto
Set a range of dBm values and the power is set automatically.
Transmit Power: 14 dBm
SSIDs: Tunnel | Bridge | [✓ Manual]



□ High_Throughput_Data_Fortinet_Default
☑ RTLS_Presence_Fortinet_Default
□ Voice_Enterprise_Fortinet_Default

AP Profiles Custom (0)

AP Profiles (76)

□ 11ac-only
□ 11n-only
□ AP-11N-default
□ FAP112B-default
□ FAP112D-default

LAN Configuration

Port ESL Mode: [✓ Offline] | NAT to LAN | Bridge to LAN | Bridge to SSID
Handoff STA Thresh: 55
Port Mode: [✓ Uplink Only] | Uplink & Bridge

ESL SES Dongle Configuration

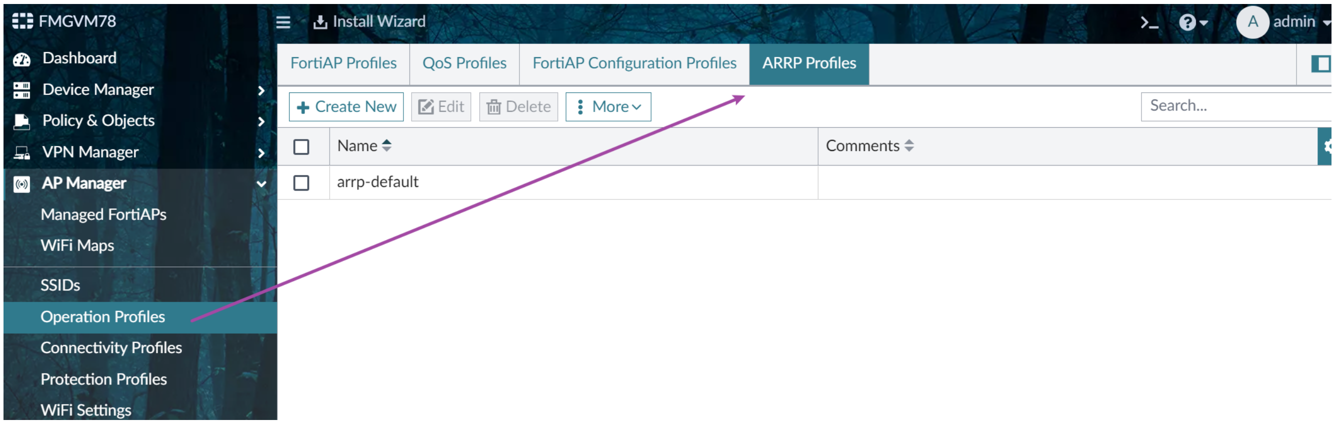
APC FQDN:

Location Based Services

FortiPresence
Mode: [✓ Disabled] | Foreign Channels Only | Foreign and Home Channels
Ekahau Blink: off
AeroScout: off
Locate WiFi Clients When Not Connected: on

ARRP profile

ARRP Profiles have been relocated from Protection Profiles to Operation Profiles.



□	Name	Comments
□	arrp-default	

Routing

7.4.0

- Using MP-BGP EVPN with VXLAN on page 119
- Add route tag address objects on page 130
- Allow better control over the source IP used by each egress interface for local out traffic on page 132
- BGP conditional advertisements for IPv6 prefix when IPv4 prefix conditions are met and vice-versa on page 140

7.4.1

- SD-WAN multi-PoP multi-hub large scale design and failover 7.4.1 on page 145

7.4.2

- Support IPsec tunnel to change names 7.4.2 on page 164
- Enhance IPv6 VRRP state control 7.4.2 on page 167
- SD-WAN hub and spoke speed test improvements 7.4.2 on page 169

Using MP-BGP EVPN with VXLAN

This information is also available in the FortiOS 7.4 Administration Guide:

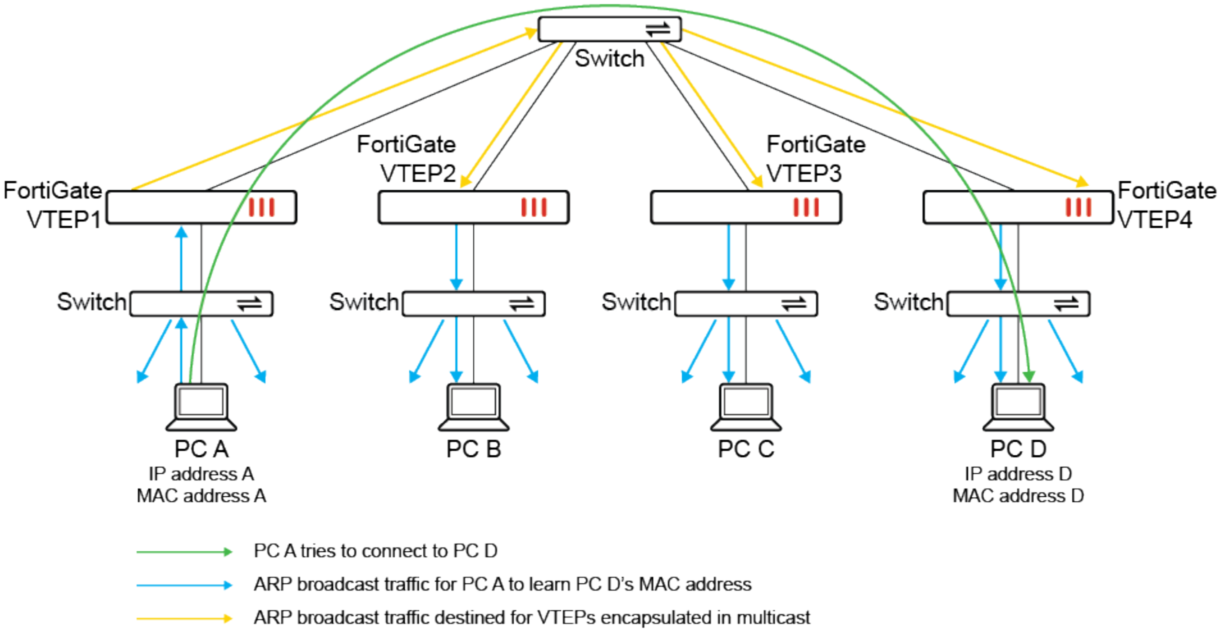
- VXLAN with MP-BGP EVPN

FortiOS supports VXLAN as implemented according to RFC 7348. Currently, VXLAN relies on determining the MAC address of the destination host by using address resolution protocol (ARP) broadcast frames encapsulated in multicast packets.

- A multicast group is maintained with all the VXLAN tunnel endpoints (VTEPs) associated with the same VXLAN, namely, with the same VXLAN network identifier (VNI).
- The multicast packets that encapsulate ARP broadcast frames are sent to this multicast group, and then the destination host replies to the source host using unicast IP packet encapsulated using VXLAN.
- The source and destination FortiGates as VTEPs each maintain a mapping of MAC addresses to remote VTEPs.

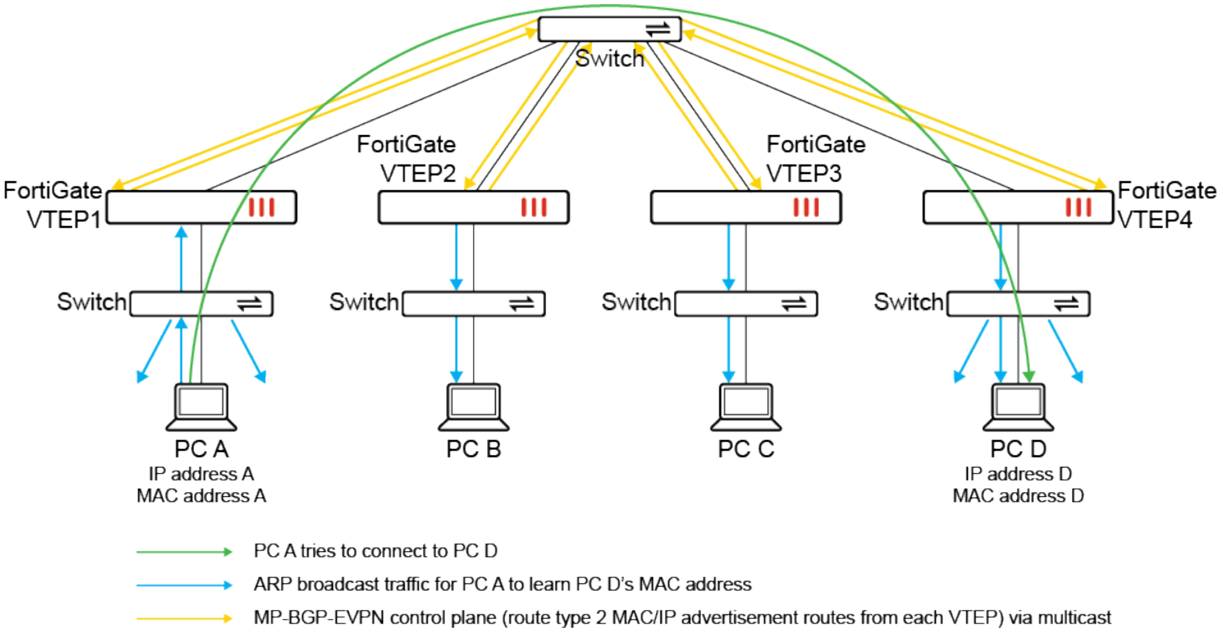
As with non-VXLAN traffic, VXLAN relies on the preceding ARP process, commonly known as flood-and-learn that floods the network with broadcast frames encapsulated as multicast packets to learn MAC addresses. In the RFC 7348 implementation of VXLAN, the data plane is simultaneously used as a control plane.

The following topology demonstrates how flood-and-learn uses ARP broadcast traffic flooded throughout the VXLAN for PC A to learn PC D's MAC address when PC A tries to connect to PC D.



In FortiOS 7.4.0, Multiprotocol Border Gateway Protocol Ethernet Virtual Private Network (MP-BGP EVPN) support for VXLAN allows for learning MAC addresses in a way that is more suitable for large deployments than flood-and-learn.

MP-BGP EVPN is a standards-based control plane that supports the distribution of attached host MAC and IP addresses using MP-BGP, namely, using the EVPN address family and MAC addresses treated as routing entries in BGP. As a control plane that is separate from the data plane, MP-BGP EVPN avoids flood-and-learn in the network, and the wide use of BGP as an external gateway protocol on the internet proves its ability to scale well with large deployments. The following topology demonstrates how MP-BGP EVPN distributes route type 2 MAC/IP advertisement routes among VTEPs in the VXLAN, and minimizes ARP broadcast traffic required for PC A to learn PC D's MAC address when PC A tries to connect to PC D.


Switch
FortiGate VTEP1
FortiGate VTEP2
FortiGate VTEP3
FortiGate VTEP4
Switch
Switch
Switch
Switch
PC A
IP address A
MAC address A
PC B
PC C
PC D
IP address D
MAC address D
PC A tries to connect to PC D
ARP broadcast traffic for PC A to learn PC D's MAC address
MP-BGP-EVPN control plane (route type 2 MAC/IP advertisement routes from each VTEP) via multicast

MP-BGP EVPN supports the following features:

- Route type 2 (MAC/IP advertisement route) and route type 3 (inclusive multicast Ethernet tag route)
- Intra-subnet communication
- Single-homing use cases
- VLAN-based service, namely, there is only one broadcast domain per EVPN instance (EVI). This is due to the current VXLAN design that supports a single VNI for a VXLAN interface.
- EVPN running on IPv4 unicast VXLAN
- Egress replication for broadcast, unknown unicast, and multicast (BUM) traffic
- VXLAN MAC learning from traffic
- IP address local learning
- ARP suppression

For more information about MP-BGP EVPN, see RFC 7432. For more information about EVPN and VXLAN, see RFC 8365.

Basic MP-BGP EVPN configuration

The MP-BGP EVPN feature builds on the CLI commands used for configuring VXLAN using a VXLAN tunnel endpoint (VTEP). See General VXLAN configuration and topologies in the FortiOS Administration Guide for more details.

After configuring VXLAN using a VTEP, the following CLI commands are configured to enable MP-BGP EVPN on each VTEP.

To configure MP-BGP EVPN on each VTEP:

1. Configure the EVPN settings:

```
config system evpn
    edit <id>
        set rd {AA | AA:NN | A.B.C.D:NN}
        set import-rt <AA:NN>
        set export-rt <AA:NN>
        set ip-local-learning {enable | disable}
        set arp-suppression {enable | disable}
    next
end
```

The `ip-local-learning` setting is used to enable/disable monitoring the local ARP table of the switch interface to learn the IP/MAC bindings, and advertise them to neighbors. This setting is disabled by default, but must be enabled when configuring MP-BGP EVPN.

The `arp-suppression` setting is used to enable/disable using proxy ARP to perform suppression of ARP discovery using the flood-and-learn approach. This setting is disabled by default. When enabled, proxy ARP entries are added on the switch interface to suppress the ARP flooding of known IP/MAC bindings, which were learned by the MP-BGP EVPN control plane.

2. Configure the EVPN settings within the VXLAN settings:

```
config system vxlan
    edit <name>
        set interface <string>
```

```
            set vni <integer>
            set evpn-id <integer>
            set learn-from-traffic {enable | disable}
        next
    end
```

The `learn-from-traffic` setting is used to enable/disable learning of remote VNIs from VXLAN traffic. This setting is disabled by default, and should only be enabled when local and all remote peers are using same VNI value, and some of the peers do not have MP-BGP EVPN capability.

3. Configure the BGP settings:

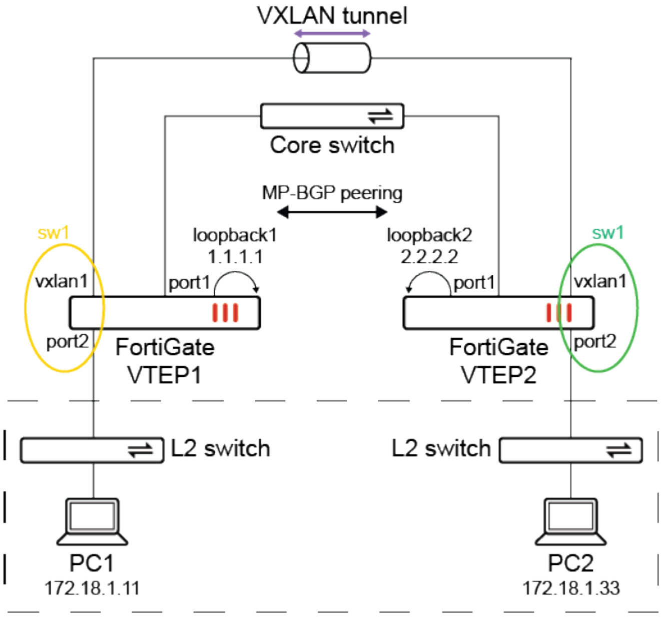
```
config router bgp
    set ibgp-multipath {enable | disable}
    set recursive-next-hop {enable | disable}
    set graceful-restart {enable | disable}
    config neighbor
        edit <WAN_IP_of_other_VTEP>
            set ebgp-enforce-multihop {enable | disable}
            set next-hop-self {enable | disable}
            set next-hop-self-vpnv4 {enable | disable}
            set soft-reconfiguration {enable | disable}
            set soft-reconfiguration-evpn {enable | disable}
            set remote-as <AS_number>
        next
    end
end
```

4. Configure the EVPN setting within the HA settings:

```
config system ha
    set evpn-ttl <integer>
end
```

Example

In this example, two FortiGates are configured as VXLAN tunnel endpoints (VTEPs). A VXLAN is configured to allow L2 connectivity between the networks behind each FortiGate. The VXLAN interface vxlan1 and port2 are placed on the same L2 network using a software switch (sw1). An L2 network is formed between PC1 and PC2. MP-BGP EVPN is used as the control plane to learn and distribute MAC address information within a single L2 domain identified using a specific VNI.



The VTEPs have the following MAC address tables:

Interface/endpoint	VTEP1	VTEP2
vxlan1	82:51:d1:44:bf:93	d2:21:00:c9:e6:98
port2	50:00:00:03:00:01	50:00:00:04:00:01
sw1	50:00:00:03:00:01	50:00:00:04:00:01

The MAC address of PC1 is 00:50:00:00:06:00. The MAC address of PC2 is 00:50:00:00:07:00.

This example assumes that the WAN interface and default route settings have already been configured on the VTEP 1 and VTEP 2 FortiGates. These configurations are omitted from the example. All peers are configured for MP-BGP EVPN.

To configure the VTEP1 FortiGate:

1. Configure the loopback interface:

```
config system interface
    edit "loopback1"
        set vdom "root"
        set ip 1.1.1.1 255.255.255.255
        set allowaccess ping https ssh http
        set type loopback
    next
end
```

2. Configure the EVPN settings:

```
config system evpn
    edit 100
        set rd "100:100"
        set import-rt "1:1"
        set export-rt "1:1"
        set ip-local-learning enable
        set arp-suppression enable
    next
end
```

3. Configure the local interface and EVPN settings within the VXLAN settings:

```
config system vxlan
    edit "vxlan1"
        set interface "loopback1"
        set vni 1000
        set evpn-id 100
    next
end
```

4. Configure the EVPN settings within the BGP settings:

```
config router bgp
    set as 65001
    set router-id 1.1.1.1
    set ibgp-multipath enable
    set recursive-next-hop enable
    set graceful-restart enable
    config neighbor
        edit "172.25.160.101"
            set ebgp-enforce-multihop enable
            set next-hop-self enable
            set next-hop-self-vpnv4 enable
            set soft-reconfiguration enable
            set soft-reconfiguration-evpn enable
            set remote-as 65001
        next
    end
    config network
        edit 1
            set prefix 1.1.1.1 255.255.255.255
        next
    end
end
```

172.27.16.237 is the WAN IP address of the VTEP2 FortiGate.

5. Configure the software switch:

```
config system switch-interface
    edit "sw1"
        set vdom "root"
        set member "port2" "vxlan1"
        set intra-switch-policy explicit
    next
end
```

6. Configure the software switch interface settings:

```
config system interface
    edit "sw1"
        set vdom "root"
        set ip 172.18.1.253 255.255.255.0
        set allowaccess ping
        set type switch
    next
end
```

7. Configure the firewall policies between the member interfaces in the software switch:

```
config firewall policy
     edit 1
         set srcintf "port2"
         set dstintf "vxlan1"
         set action accept
         set srcaddr "all"
         set dstaddr "all"
         set schedule "always"
         set service "ALL"
     next
     edit 2
         set srcintf "vxlan1"
         set dstintf "port2"
         set action accept
         set srcaddr "all"
         set dstaddr "all"
         set schedule "always"
         set service "ALL"
     next
end
```

To configure the VTEP2 FortiGate:

1. Configure the loopback interface:

```
config system interface
    edit "loopback2"
        set vdom "root"
        set ip 2.2.2.2 255.255.255.255
        set allowaccess ping https ssh http
        set type loopback
    next
end
```

2. Configure the EVPN settings:

```
config system evpn
    edit 100
        set rd "100:100"
        set import-rt "1:1"
        set export-rt "1:1"
        set ip-local-learning enable
        set arp-suppression enable
    next
end
```

3. Configure the local interface and EVPN settings within the VXLAN settings:

```
config system vxlan
    edit "vxlan1"
        set interface "loopback2"
        set vni 1000
        set evpn-id 100
    next
end
```

4. Configure the EVPN settings within the BGP settings:

```
config router bgp
    set as 65001
    set router-id 2.2.2.2
    set ibgp-multipath enable
    set recursive-next-hop enable
    set graceful-restart enable
    config neighbor
        edit "172.25.160.100"
            set ebgp-enforce-multihop enable
            set next-hop-self enable
            set next-hop-self-vpnv4 enable
            set soft-reconfiguration enable
            set soft-reconfiguration-evpn enable
            set remote-as 65001
        next
    end
    config network
        edit 1
            set prefix 2.2.2.2 255.255.255.255
        next
    end
end
```

172.27.16.236 is the WAN IP address of the VTEP1 FortiGate.

5. Configure the software switch:

```
config system switch-interface
    edit "sw1"
        set vdom "root"
        set member "port2" "vxlan1"
        set intra-switch-policy explicit
    next
end
```

6. Configure the software switch interface settings:

```
config system interface
    edit "sw1"
        set vdom "root"
        set ip 172.18.1.254 255.255.255.0
        set allowaccess ping
        set type switch
    next
end
```

7. Configure the firewall policies between the member interfaces in the software switch:

```
config firewall policy
     edit 1
         set srcintf "port2"
         set dstintf "vxlan1"
         set action accept
```

```
        set srcaddr "all"
        set dstaddr "all"
        set schedule "always"
        set service "ALL"
    next
    edit 2
        set srcintf "vxlan1"
        set dstintf "port2"
        set action accept
        set srcaddr "all"
        set dstaddr "all"
        set schedule "always"
        set service "ALL"
    next
end
```

To verify the MP-BGP EVPN status on the VTEP1 FortiGate:

1. From a host computer with IP address 172.18.1.11, perform the following.
   1. Check the ARP cache:

      ```
      # arp
      Address                  HWtype  HWaddress           Flags Mask            Iface
      172.18.1.253             ether   50:00:00:03:00:01   C                     ens3
      ```

   2. Ping the host computer with IP address 172.18.1.33:

      ```
      # ping 172.18.1.33 -c 4
      PING 172.18.1.33 (172.18.1.33) 56(84) bytes of data.
      64 bytes from 172.18.1.33: icmp_seq=1 ttl=64 time=1325 ms
      64 bytes from 172.18.1.33: icmp_seq=2 ttl=64 time=319 ms
      64 bytes from 172.18.1.33: icmp_seq=3 ttl=64 time=3.96 ms
      64 bytes from 172.18.1.33: icmp_seq=4 ttl=64 time=1.66 ms

      --- 172.18.1.33 ping statistics ---
      4 packets transmitted, 4 received, 0% packet loss, time 3007ms
      rtt min/avg/max/mdev = 1.660/412.614/1325.209/542.530 ms
      ```

   3. Check the ARP cache again:

      ```
      # arp
      Address                  HWtype  HWaddress           Flags Mask            Iface
      172.18.1.33              ether   00:50:00:00:07:00   C                     ens3
      172.18.1.253             ether   50:00:00:03:00:01   C                     ens3
      ```

2. On the VTEP1 FortiGate, run the switch and VXLAN debug commands.
   1. Verify the forwarding database for vxlan1:

      ```
      # diagnose sys vxlan fdb list vxlan1
      mac=00:00:00:00:00:00 state=0x0082 remote_ip=2.2.2.2 port=4789 vni=1000 ifindex0
      mac=00:50:00:00:07:00 state=0x0082 remote_ip=2.2.2.2 port=4789 vni=1000 ifindex0

      total fdb num: 2
      ```

   2. Verify the forwarding database statistics for vxlan1:

      ```
      # diagnose sys vxlan fdb stat vxlan1
      fdb_table_size=256 fdb_table_used=2 fdb_entry=2 fdb_max_depth=1 cleanup_idx=0 c2
      ```

c. Verify the bridging information for sw1:

```
# diagnose netlink brctl name host sw1
show bridge control interface sw1 host.
fdb: hash size=32768, used=5, num=5, depth=1, gc_time=4, ageing_time=3, arp-sups
Bridge sw1 host table
port no device  devname mac addr                ttl     attributes
  2     15      vxlan1  00:00:00:00:00:00       28       Hit(28)
  2     15      vxlan1  00:50:00:00:07:00       18       Hit(18)
  2     15      vxlan1  82:51:d1:44:bf:93       0       Local Static
  1     4       port2   00:50:00:00:06:00       14       Hit(14)
  1     4       port2   50:00:00:03:00:01       0       Local Static
```

3. Run the BGP EVPN commands and observe the route type 2 (MAC/IP advertisement route) and route type 3 (inclusive multicast Ethernet tag route).

   a. Verify the BGP L2 VPN EVPN summary information:

```
# get router info bgp evpn summary

VRF 0 BGP router identifier 1.1.1.1, local AS number 65001
BGP table version is 2
1 BGP AS-PATH entries
0 BGP community entries

Neighbor       V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/Pd
172.25.160.101 4      65001       9       9        1    0    0 00:04:02        3

Total number of neighbors 1
```

   b. Verify the BGP L2 VPN EVPN network information:

```
# get router info bgp evpn network
   Network          Next Hop            Metric     LocPrf Weight RouteTag Path
Route Distinguisher: 100:100 (Default for VRF 0)
*> [2][0][48][00:50:00:00:06:00][0]/72
                    1.1.1.1             0             100  32768        0 i <-/>
*> [2][0][48][00:50:00:00:06:00][32][172.18.1.11]/104
                    1.1.1.1             0             100  32768        0 i <-/>
*>i[2][0][48][00:50:00:00:07:00][0]/72
                    2.2.2.2             0             100      0        0 i <-/>
*>i[2][0][48][00:50:00:00:07:00][32][172.18.1.33]/104
                    2.2.2.2             0             100      0        0 i <-/>
*> [3][0][32][1.1.1.1]/80
                    1.1.1.1             0             100  32768        0 i <-/>
*>i[3][0][32][2.2.2.2]/80
                    2.2.2.2             0             100      0        0 i <-/>

   Network          Next Hop            Metric     LocPrf Weight RouteTag Path
Route Distinguisher: 100:100 (received from VRF 0)
*>i[2][0][48][00:50:00:00:07:00][0]/72
                    2.2.2.2             0             100      0        0 i <-/>
*>i[2][0][48][00:50:00:00:07:00][32][172.18.1.33]/104
                    2.2.2.2             0             100      0        0 i <-/>
*>i[3][0][32][2.2.2.2]/80
                    2.2.2.2             0             100      0        0 i <-/>
```

   c. Verify the BGP L2 VPN EVPN context:

```
# get router info bgp evpn context
L2VPN EVPN context for VRF 0
ID 100 vlan-based, RD is [100:100]
 Import RT: RT:1:1
 Export RT: RT:1:1
  Bridge domain 0 VNI 1000
   Encapsulation 8(VXLAN)
   Source interface loopback1
   Source address 1.1.1.1
```

d. Verify the neighbor EVPN routes:

```
# get router info bgp neighbors 172.25.160.101 routes evpn
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight RouteTag Path
Route Distinguisher: 100:100 (Default for VRF 0) (Default for VRF 0)
*>i[2][0][48][00:50:00:00:07:00][0]/72
                    2.2.2.2             0             100      0        0 i <-/>
*>i[2][0][48][00:50:00:00:07:00][32][172.18.1.33]/104
                    2.2.2.2             0             100      0        0 i <-/>
*>i[3][0][32][2.2.2.2]/80
                    2.2.2.2             0             100      0        0 i <-/>
Route Distinguisher: 100:100 (received from VRF 0) (received from VRF 0)
*>i[2][0][48][00:50:00:00:07:00][0]/72
                    2.2.2.2             0             100      0        0 i <-/>
*>i[2][0][48][00:50:00:00:07:00][32][172.18.1.33]/104
                    2.2.2.2             0             100      0        0 i <-/>
*>i[3][0][32][2.2.2.2]/80
                    2.2.2.2             0             100      0        0 i <-/>

Total number of prefixes 6
```

4. Run the following EVPN get commands.

a. Verify the EVPN instances:

```
# get l2vpn evpn instance
EVPN instance: 100
IP local learning enabled
ARP suppression enabled
HA primary
   Number of bridge domain: 1
   Bridge domain: TAGID 0 VNI 1000 ADDR 1.1.1.1 VXLAN vxlan1 SWITCH sw1
```

b. Verify the EVPN table:

```
# get l2vpn evpn table
EVPN instance 100
 Broadcast domain VNI 1000 TAGID 0

  EVPN instance 100
 Broadcast domain VNI 1000 TAGID 0

  EVPN MAC table:
  MAC               VNI      Remote Addr     Binded Address
```

```
00:50:00:00:07:00 1000     2.2.2.2         172.18.1.33
                  1000     2.2.2.2         -

EVPN IP table:
Address         VNI      Remote Addr     MAC
172.18.1.33     1000     2.2.2.2         00:50:00:00:07:00

EVPN Local MAC table:
"Inactive" means this MAC/IP pair will not be sent to peer.
Flag code: S - Static F - FDB. Trailing * means HA
MAC               Flag Status   Binded Address
00:50:00:00:06:00      Active   172.18.1.11
                  F    Active   -

EVPN Local IP table:
Address         MAC
172.18.1.11     00:50:00:00:06:00

EVPN PEER table:
VNI      Remote Addr     Binded Address
1000     2.2.2.2         2.2.2.2
```

5. Run the proxy ARP diagnose command:

   ```
   # diagnose ip parp list
   Address           Hardware Addr     Interface
   172.18.1.33       00:50:00:00:07:00 sw1
   ```

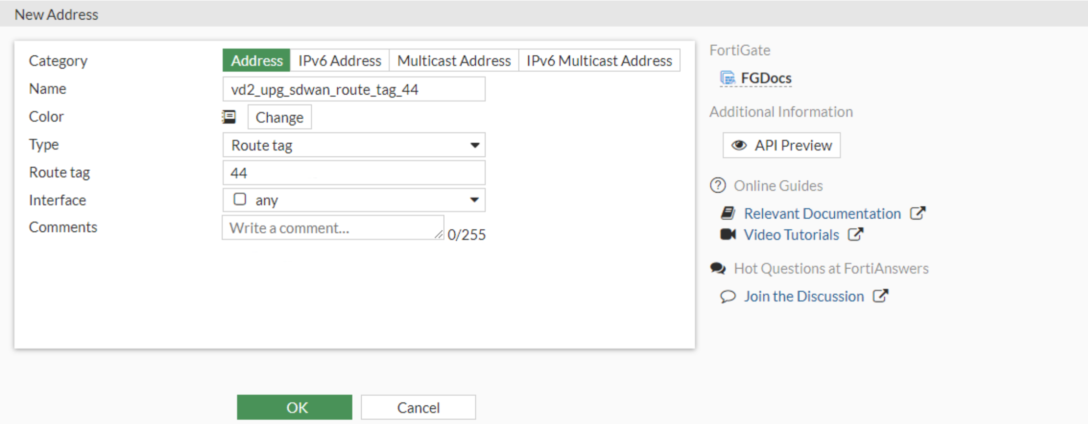
Add route tag address objects

A route tag (`route-tag`) firewall address object can include IPv4 or IPv6 addresses associated with a BGP route tag number, and is updated dynamically with BGP routing updates. The route tag firewall address object allows for a more dynamic and flexible configuration that does not require manual intervention to dynamic routing updates. This address object can be used wherever a firewall address can be used, such as in a firewall policy, a router policy, or an SD-WAN service rule.

The *Route tag* field has been removed from the *Priority Rule* configuration page (*Network > SD-WAN > SD-WAN Rules*). The `route-tag` option has been removed from the `config service settings` under `config system sdwan`.

To configure and apply a route tag address object in the GUI:

1. Configure the route tag address object:
   a. Go to *Policy & Objects > Addresses* and click *Create New > Address*.
   b. Enter a *Name*, such as *vd2_upg_sdwan_route_tag_44*.
   c. Set the *Type* to *Route tag*.
   d. Enter the *Route tag* number, such as *44*.



New Address

Category	Address | IPv6 Address | Multicast Address | IPv6 Multicast Address
Name	vd2_upg_sdwan_route_tag_44
Color	Change
Type	Route tag
Route tag	44
Interface	□ any
Comments	Write a comment... 0/255

   e. Click *OK*.

2. Add the address to a firewall policy:
   a. Go to *Policy & Objects > Firewall Policy*.
   b. Edit an existing policy or create a new one.
   c. Set the *Destination* to *vd2_upg_sdwan_route_tag_44*.
   d. Configure the other settings as needed.
   e. Click *OK*.
3. Add the address to an SD-WAN service rule:
   a. Go to *Network > SD-WAN* and select the *SD-WAN Rules* tab.
   b. Edit an existing rule or create a new one.
   c. In the *Destination* section, set the *Address* to *vd2_upg_sdwan_route_tag_44*.
   d. Configure the other settings as needed.
   e. Click *OK*.

To configure and apply a route tag address object in the CLI:

1. Configure the route tag address object:

```
config firewall address
    edit "vd2_upg_sdwan_route_tag_44"
        set type route-tag
        set route-tag 44
    next
end
```

2. Add the address to a firewall policy:

```
config firewall policy
    edit 3
        set srcintf "any"
        set dstintf "any"
        set action accept
        set srcaddr "all"
        set dstaddr "vd2_upg_sdwan_route_tag_44"
        set schedule "always"
        set service "ALL"
    next
end
```

3. Add the address to an SD-WAN service rule:

```
config system sdwan
    config service
        edit 1
            set dst "vd2_upg_sdwan_route_tag_44"
            set priority-members 1
        next
    end
end
```

To verify the configuration:

1. After some traffic passes, verify that the route tag firewall address is associated with policy ID 3:

```
# diagnose firewall iprope list | grep -A 15 index=3
policy index=3 uuid_idx=754 action=accept
flag (8010008): redir master pol_stats
flag2 (4000): resolve_sso
flag3 (100000a0): link-local best-route no-vwp
schedule(always)
cos_fwd=255  cos_rev=255
group=00100004 av=00004e20 au=00000000 split=00000000
host=5 chk_client_info=0x0 app_list=0 ips_view=0
misc=0
zone(1): 0 -> zone(1): 0
source(1): 0.0.0.0-255.255.255.255, uuid_idx=684,
service(1):
        [0:0x0:0/(0,65535)->(0,65535)] flags:0 helper:auto
 route_tag(1):  44
```

2. Verify the list of firewall route tag addresses:

```
# diagnose firewall route_tag list
list route tag info(vf(vd2)):
route tag address, route_tag(30) vrf_num(1):
vrf id(0), num(2): 11.11.11.11-11.11.11.11 100.1.1.0-100.1.1.255

route tag address, route_tag(33) vrf_num(1):
vrf id(0), num(1): 33.1.1.0-33.1.1.255

route tag address, route_tag(40) vrf_num(1):
vrf id(0), num(2): 11.11.11.11-11.11.11.11 100.1.1.0-100.1.1.255

route tag address, route_tag(44) vrf_num(1):
vrf id(0), num(1): 33.1.1.0-33.1.1.255
```

Allow better control over the source IP used by each egress interface for local out traffic

This information is also available in the FortiOS 7.4 Administration Guide:

- Defining a preferred source IP for local-out egress interfaces
- Defining a preferred source IP for local-out egress interfaces on BGP routes
- Defining a preferred source IP for local-out egress interfaces on SD-WAN members

Better control over the source IP used by each egress interface is feasible by allowing a preferred source IP to be defined in each of these scenarios.

- Configuring a static route:

```
config router static
    edit <id>
        set preferred-source <ip_address>
    next
end
```

- Configuring an SD-WAN member:

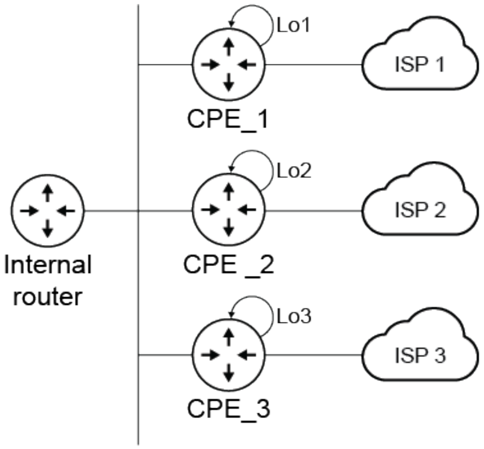
```
config system sdwan
    config members
        edit <id>
            set preferred-source <ip_address>
        next
    end
end
```

- Configuring a route map so that a BGP route can support a preferred source:

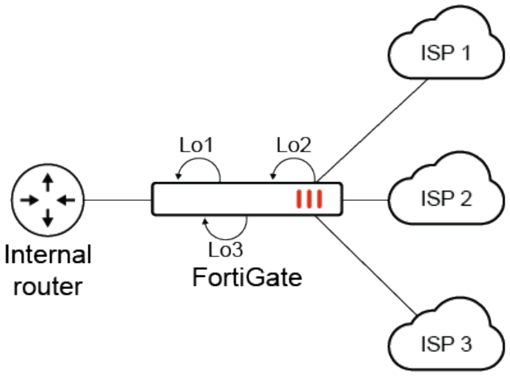
```
config router route-map
    edit <name>
        config rule
            edit <id>
                set set-ip-prefsrc <ip_address>
            next
        end
    next
end
```

Example configurations

In scenarios where multiple CPE (customer premise equipment) routers are used for each transport, it is easy to define a public IP per router as a loopback IP. Then, locally sourced traffic and BGP routes can use the public loopback IP as source.



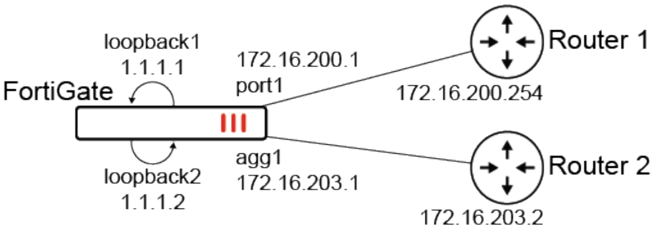
When a FortiGate is used to replace multiple CPE routers, it must be able to source traffic with the public IP assigned by their respective ISP that is assigned to the loopback interfaces.



This feature allows the preferred source IP to be configured in the following scenarios so that local out traffic is sourced from these IPs.

Example 1

In this example, a source IP is defined per static route. Local traffic that uses the static route will use the source IP instead of the interface IP associated with the route.



To configure preferred source IPs for static routes:

1. Configure the static routes:

```
config router static
    edit 22
        set dst 172.17.254.0 255.255.255.0
        set gateway 172.16.200.254
        set preferred-source 1.1.1.1
        set distance 2
        set device "port1"
    next
    edit 23
        set dst 172.17.254.0 255.255.255.0
        set gateway 172.16.203.2
        set preferred-source 1.1.1.2
        set distance 2
        set device "agg1"
    next
end
```

2. Configure the primary DNS server IP address:

```
config system dns
    set primary 172.17.254.148
end
```

To verify the configuration:

1. Verify the kernel routing table:

```
# get router info kernel
...
tab=254 vf=0 scope=0 type=1 proto=11 prio=1 0.0.0.0/0.0.0.0/0->172.17.254.0/24
pref=0.0.0.0
        gwy=172.16.200.254 flag=14 hops=0 oif=9(port1) pref=1.1.1.1
        gwy=172.16.203.2 flag=14 hops=0 oif=33(agg1) pref=1.1.1.2
```

2. Verify the routing table for 172.17.254.148:

```
# get router info routing-table details 172.17.254.148
Routing table for VRF=0
Routing entry for 172.17.254.0/24
  Known via "static", distance 2, metric 0, best
  * vrf 0 172.16.200.254, via port1, prefsrc 1.1.1.1
  * vrf 0 172.16.203.2, via agg1, prefsrc 1.1.1.2
```

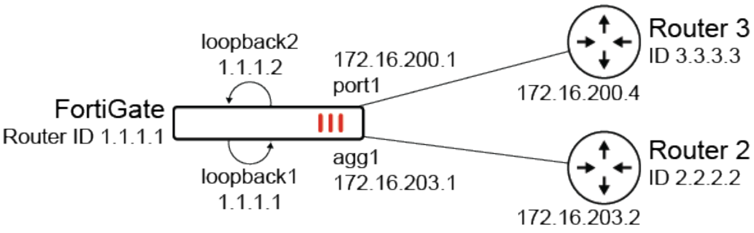
3. Run a sniffer trace after some traffic passes:

```
# diagnose sniffer packet any "host 172.17.254.148" 4
interfaces=[any]
filters=[host 172.17.254.148]
1.319811 port1 out 1.1.1.1.1371 -> 172.17.254.148.53: udp 43
1.320095 port1 in 172.17.254.148.53 -> 1.1.1.1.1371: udp 310
1.921718 port1 out 1.1.1.1.1371 -> 172.17.254.148.53: udp 27
2.031520 port1 in 172.17.254.148.53 -> 1.1.1.1.1371: udp 213
```

When DNS traffic leaves the FortiGate and is routed through port1, the source address 1.1.1.1 is used.

Example 2:

In this example, a route map is configured to set the preferred source IP so that the BGP route can support the preferred source.



To configure preferred source IPs for BGP routing:

1. Configure the route maps:

```
config router route-map
    edit "map1"
        config rule
```

```
            edit 1
                set set-ip-prefsrc 1.1.1.1
            next
        end
    next
    edit "map2"
        config rule
            edit 1
                set set-ip-prefsrc 1.1.1.2
            next
        end
    next
end
```

2. Configure the BGP settings:

```
config router bgp
    set as 65412
    set router-id 1.1.1.1
    set ibgp-multipath enable
    set cluster-id 1.1.1.1
    set graceful-restart enable
    config aggregate-address
        edit 1
            set prefix 172.28.0.0 255.255.0.0
            set as-set enable
            set summary-only enable
        next
    end
    config neighbor
        edit "3.3.3.3"
            set capability-graceful-restart enable
            set soft-reconfiguration enable
            set prefix-list-out "local-out"
            set remote-as 65412
            set route-map-in "map2"
            set route-map-out "as-prepend"
            set keep-alive-timer 30
            set holdtime-timer 90
            set update-source "loopback1"
            set route-reflector-client enable
        next
        edit "2.2.2.2"
            set advertisement-interval 5
            set activate6 disable
            set capability-graceful-restart enable
            set soft-reconfiguration enable
            set distribute-list-out "local-out-FGTB-deny"
            set remote-as 65412
            set route-map-in "map1"
            set route-map-out "as-rewrite"
            set keep-alive-timer 30
            set holdtime-timer 90
            set update-source "loopback1"
        next
    end
end
```

To verify the configuration:

1. Verify the BGP routing table for 172.25.1.0/24:

   ```
   # get router info bgp network 172.25.1.0/24
   VRF 0 BGP routing table entry for 172.25.1.0/24
   Paths: (1 available, best #1, table Default-IP-Routing-Table)
     Not advertised to any peer
     Original VRF 0
     Local
       2.2.2.2 (metric 10050) from 2.2.2.2 (2.2.2.2)
         Origin IGP metric 0, localpref 100, valid, internal, best, prefsrc 1.1.1.1
         Last update: Wed Jan 25 15:15:48 2023
   ```

2. Verify the BGP routing table for 172.28.5.0/24:

   ```
   # get router info bgp network 172.28.5.0/24
   VRF 0 BGP routing table entry for 172.28.5.0/24
   Paths: (1 available, best #1, table Default-IP-Routing-Table, Advertisements suppressed
   by an aggregate.)
     Not advertised to any peer
     Original VRF 0
     65050, (Received from a RR-client)
       3.3.3.3 (metric 11000) from 3.3.3.3 (3.3.3.3)
         Origin IGP metric 0, localpref 100, valid, internal, best, prefsrc 1.1.1.2
         Last update: Wed Jan 25 15:15:48 2023
   ```

3. Verify the kernel routing table for 172.28.5.0/24:

   ```
   # get router info kernel | grep -B 2 172.28.5.0/24
   tab=254 vf=0 scope=0 type=1 proto=11 prio=1 0.0.0.0/0.0.0.0/0->172.28.1.0/24
   pref=1.1.1.2 gwy=172.16.200.4 dev=9(port1)
   tab=254 vf=0 scope=0 type=1 proto=11 prio=1 0.0.0.0/0.0.0.0/0->172.28.2.0/24
   pref=1.1.1.2 gwy=172.16.200.4 dev=9(port1)
   tab=254 vf=0 scope=0 type=1 proto=11 prio=1 0.0.0.0/0.0.0.0/0->172.28.5.0/24
   pref=1.1.1.2 gwy=172.16.200.4 dev=9(port1)
   ```

4. Verify the kernel routing table for 172.25.1.0/24:

   ```
   # get router info kernel | grep -A 2 172.25.1.0/24
   tab=254 vf=0 scope=0 type=1 proto=11 prio=1 0.0.0.0/0.0.0.0/0->172.25.1.0/24
   pref=1.1.1.1 gwy=172.16.203.2 dev=33(agg1)
   tab=254 vf=0 scope=0 type=1 proto=11 prio=1 0.0.0.0/0.0.0.0/0->172.26.1.0/24
   pref=1.1.1.1 gwy=172.16.203.2 dev=33(agg1)
   tab=254 vf=0 scope=0 type=1 proto=11 prio=1 0.0.0.0/0.0.0.0/0->172.26.2.0/24
   pref=1.1.1.1 gwy=172.16.203.2 dev=33(agg1)
   ```

   The FortiGate learns routes from router 3.3.3.3 and prefers the source IP of 1.1.1.2. It learns routes from router 2.2.2.2 and prefers source IP of 1.1.1.1.

5. Run a sniffer trace after some traffic passes.

   a. When trying to reach a destination in the 172.25.1.0/0 subnet through router 2.2.2.2:

      ```
      # diagnose sniffer packet any "icmp" 4
      interfaces=[any]
      filters=[icmp]
      9.244334 agg1 out 1.1.1.1 -> 172.25.1.2: icmp: echo request
      9.244337 port12 out 1.1.1.1 -> 172.25.1.2: icmp: echo request
      ```

```
10.244355 agg1 out 1.1.1.1 -> 172.25.1.2: icmp: echo request
10.244357 port12 out 1.1.1.1 -> 172.25.1.2: icmp: echo request
```

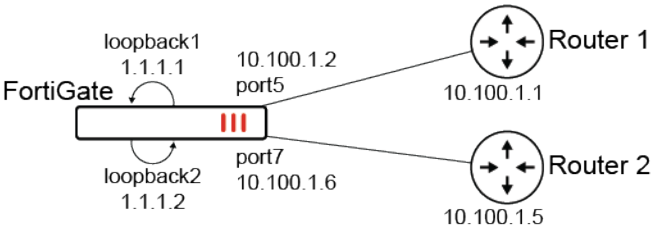
b. When trying to reach a destination in the 172.28.5.0/24 subnet through router 3.3.3.3:

```
# diagnose sniffer packet any "icmp" 4
interfaces=[any]
filters=[icmp]
2.434035 port1 out 1.1.1.2 -> 172.28.5.2: icmp: echo request
3.434059 port1 out 1.1.1.2 -> 172.28.5.2: icmp: echo request
```

Traffic destined for the 172.25.1.0/24 subnet uses 1.1.1.1 as source. Traffic destined for the 172.28.5.0/24 subnet uses 1.1.1.2 as source.

Example 3:

In this example, two SD-WAN members, port5 and port6, will use loopback1 and loopback2 as sources instead of their physical interface address. A static route is created for destination 200.0.0.0/24 to use the virtual-wan-link. In turn, the FortiGate will create two ECMP routes to the member gateways and source the traffic from the loopback IPs.



To configure preferred source IPs for SD-WAN members:

1. Configure the SD-WAN members and other settings:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "port5"
            set gateway 10.100.1.1
            set preferred-source 1.1.1.1
            set source 1.1.1.1
        next
        edit 2
            set interface "port7"
            set gateway 10.100.1.5
            set preferred-source 1.1.1.2
            set source 1.1.1.2
        next
    end
end
```

In the SD-WAN `config members` settings, configuring the `source` for the health check probes is still required. SD-WAN adds dedicated kernel routes (proto=17) for the health checks using the interface IP or source IP when specified. To view the kernel routes, use `diagnose ip route list`.

2. Configure the static route:

```
config router static
    edit 2000
        set dst 200.0.0.0 255.255.255.0
        set distance 1
        set sdwan-zone "virtual-wan-link"
    next
end
```

To verify the configuration:

1. Verify the kernel routing table for 200.0.0.0/24:

```
# get router info kernel | grep -A 2 200.0.0.0/24
tab=254 vf=0 scope=0 type=1 proto=11 prio=1 0.0.0.0/0.0.0.0/0->200.0.0.0/24 pref=0.0.0.0
        gwy=10.100.1.1 flag=14 hops=255 oif=13(port5) pref=1.1.1.1
        gwy=10.100.1.5 flag=14 hops=254 oif=15(port7) pref=1.1.1.2
```

2. Verify the routing table for 200.0.0.0/24:

```
# get router info routing-table details 200.0.0.0/24
Routing table for VRF=0
Routing entry for 200.0.0.0/24
  Known via "static", distance 1, metric 0, best
  * vrf 0 10.100.1.1, via port5, prefsrc 1.1.1.1
  * vrf 0 10.100.1.5, via port7, prefsrc 1.1.1.2
```

3. Run a sniffer trace after some traffic passes.
   a. When traffic leaves port5:

```
# diagnose sniffer packet any "host 200.0.0.1" 4
interfaces=[any]
filters=[host 200.0.0.1]
6.592488 port5 out 1.1.1.1 -> 200.0.0.1: icmp: echo request
7.592516 port5 out 1.1.1.1 -> 200.0.0.1: icmp: echo request
8.592532 port5 out 1.1.1.1 -> 200.0.0.1: icmp: echo request
```

   b. When traffic leaves port7:

```
# diagnose sniffer packet any "host 200.0.0.1" 4
interfaces=[any]
filters=[host 200.0.0.1]
75.664173 port7 out 1.1.1.2 -> 200.0.0.1: icmp: echo request
76.664194 port7 out 1.1.1.2 -> 200.0.0.1: icmp: echo request
```

Traffic exiting each interface is sourced from the corresponding loopback IP.

BGP conditional advertisements for IPv6 prefix when IPv4 prefix conditions are met and vice-versa

This information is also available in the FortiOS 7.4 Administration Guide:

- BGP conditional advertisements for IPv6 prefix when IPv4 prefix conditions are met and vice-versa

BGP conditional advertisement allows the router to advertise a route only when certain conditions are met. Multiple conditions can be used together, with conditional route map entries treated as an AND operator. The FortiGate supports conditional advertisement of IPv4 and IPv6 route maps with `edit <advertise-routemap>` under `config conditional-advertise`, and supports configuring IPv4 and IPv6 route maps as conditions with the `condition-routemap` setting.

The FortiGate can cross-check conditions involving IPv4 and IPv6 route maps and perform conditional advertisements accordingly when those conditions are met. The global option, `cross-family-conditional-adv` in the BGP configuration settings allows this cross-checking to occur.

```
config router bgp
    set cross-family-conditional-adv {enable | disable}
    config conditional-advertise
        edit <advertise-routemap>
            set advertise-routemap <string>
            set condition-routemap <name1>, <name2>, ...
            set condition-type {exist | non-exist}
        next
    end
end
```

By default, the `cross-family-conditional-adv` setting is disabled. When disabled, the FortiGate will only check conditional route maps against the routing information base (RIB) of the IP address family (IPv4 or IPv6) that corresponds to the IP address family of the route map to be advertised conditionally.

For example, for an IPv6 conditional advertisement, if IPv4 conditional route maps have been configured, then the FortiGate will not meet any of these conditions because IPv4 routes will not exist in the IPv6 RIB. The same behavior applies for an IPv4 conditional advertisement, namely, that the FortiGate will not meet any configured IPv6 conditions since these routes will not exist in the IPv4 RIB. If routes do not match a conditional route map, then the condition is considered non-existent.

IPv4 and IPv6 BGP conditional advertisements using advertising and conditional route maps of the same IP address family are already supported in previous versions of FortiOS.

DS-Lite example

In this example, the FortiGate acts as a Dual-Stack Lite (DS-Lite) address family transition router (AFTR) where the customer equipment (CE) network via Router1 uses IPv6 and where Router2 is the internet gateway using IPv4.

The administrator of the AFTR has the following requirements:

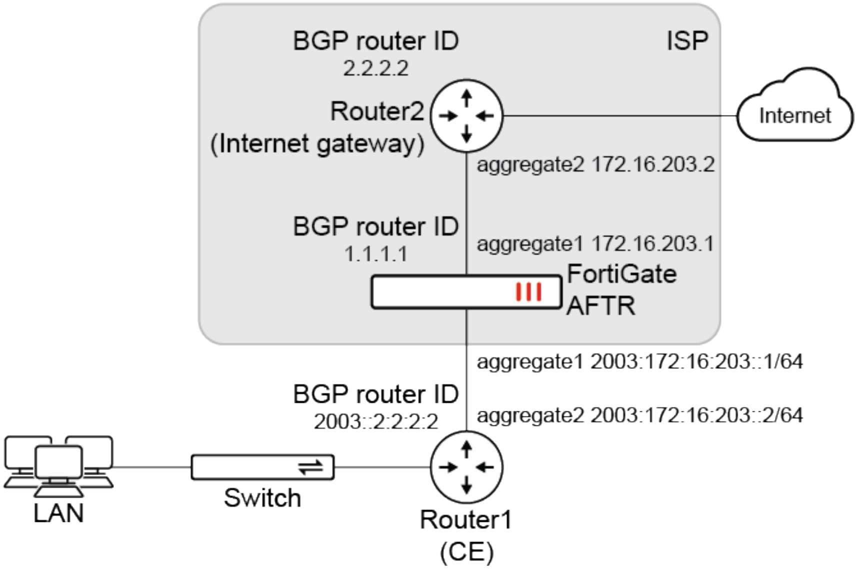
- The FortiGate needs to announce IPv4 pools for NAT translation towards the internet gateway only if the IPv6 B4 prefix exists in the routing table.

- The FortiGate needs to advertise the DS-Lite termination IPv6 address towards the CE network only if the IPv4 default route exists on the FortiGate.

The prefixes defined in IPv4 route map `2814` and IPv6 route map `map-281` both exist, so the FortiGate advertises the route map prefix in route-map `2224` (172.22.2.0/255.255.255.0) to its BGP neighbor 2.2.2.2.

For IPv6 neighbor 2003::2:2:2:2, the prefixes defined in IPv4 route map `2874` and IPv6 route map `map-38` both do not exist, and the `condition-type` is set to `non-exist`, so the FortiGate advertises the route map prefix in route map `map-222` (2003:172:22:1::/64) to its BGP neighbor 2003::2:2:2:2.

When the global `cross-family-conditional-adv` enabled, this is the only time the FortiGate will cross-check the address family; otherwise, it only checks the corresponding conditional map and treats the cross-family addresses as non-existent.



To configure the BGP settings with address family cross-checking:

```
config router bgp
    set as 65412
    set router-id 1.1.1.1
    set ibgp-multipath enable
    set network-import-check disable
    set cluster-id 1.1.1.1
    set graceful-restart enable
    set cross-family-conditional-adv enable
    config neighbor
        edit "3.3.3.3"
            set activate6 disable
            set capability-graceful-restart enable
            set soft-reconfiguration enable
            set prefix-list-out "local-out"
            set remote-as 65412
            set route-map-out "as-prepend"
            set keep-alive-timer 30
            set holdtime-timer 90
            set update-source "loopback1"
            set route-reflector-client enable
        next
        edit "2.2.2.2"
            set advertisement-interval 5
            set activate6 disable
            set capability-graceful-restart enable
            set soft-reconfiguration enable
            set remote-as 65412
            set keep-alive-timer 34
            set holdtime-timer 90
            set update-source "loopback1"
            config conditional-advertise
                edit "2224"
                    set condition-routemap "2814" "map-281"
                next
            end
            set route-reflector-client enable
        next
        edit "2003::2:2:2:2"
            set advertisement-interval 5
            set activate disable
            set capability-graceful-restart6 enable
            set soft-reconfiguration enable
            set soft-reconfiguration6 enable
            set remote-as 65412
            set keep-alive-timer 30
            set holdtime-timer 90
            set update-source "loopback1"
            config conditional-advertise6
                edit "map-222"
                    set condition-routemap "map-38" "2874"
                    set condition-type non-exist
                next
            end
            set route-reflector-client6 enable
        next
        edit "2003::3:3:3:3"
            set advertisement-interval 5
            set activate disable
            set capability-graceful-restart6 enable
            set soft-reconfiguration6 enable
            set remote-as 65412
            set route-map-in6 "community-del777"
            set keep-alive-timer 30
            set holdtime-timer 90
            set update-source "loopback1"
        next
    end
    config network
        edit 1
            set prefix 172.27.1.0 255.255.255.0
        next
        edit 2
            set prefix 172.27.2.0 255.255.255.0
        next
        edit 3
            set prefix 172.22.2.0 255.255.255.0
        next
    end
    config network6
        edit 1
            set prefix6 2003:172:22:1::/64
        next
    end
end
```

To verify the BGP status and the BGP routing table for IPv4:

```
# get router info bgp summary
VRF 0 BGP router identifier 1.1.1.1, local AS number 65412
BGP table version is 2
6 BGP AS-PATH entries
2 BGP community entries

Neighbor   V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2.2.2.2    4      65412     100     148        2    0    0 00:42:22        3
3.3.3.3    4      65412      99      99        2    0    0 00:42:05        6
6.6.6.6    4         20       0       0        0    0    0    never Idle (Admin)
10.100.1.1 4         20     100     107        2    0    0 00:43:43        2
10.100.1.5 4         20      53      57        2    0    0 00:43:42        0

Total number of neighbors 5

Condition route map:
  2814, state 1, use 3
  map-281, state 1, use 3
```

To verify the BGP status and the BGP routing table for IPv6:

```
# get router info6 bgp summary
VRF 0 BGP router identifier 1.1.1.1, local AS number 65412
BGP table version is 3
6 BGP AS-PATH entries
2 BGP community entries

Neighbor      V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
6.6.6.6       4         20       0       0        0    0    0    never Idle (Admin)
10.100.1.1    4         20     100     108        3    0    0 00:43:51        0
10.100.1.5    4         20      53      57        3    0    0 00:43:50        0
2003::2:2:2:2 4      65412      98     118        3    0    0 00:42:25        1
2003::3:3:3:3 4      65412     102     100        2    0    0 00:42:20        3

Total number of neighbors 5

Condition route map:
  map-38, state 0, use 3
  2874, state 0, use 3
```

To verify the BGP routing table for IPv4 and confirm the conditional advertisement occurred:

```
# get router info routing-table bgp
Routing table for VRF=0
B       172.22.2.0/24 [200/0] via 1.1.1.1 (recursive via 172.16.203.1, agg2), 00:00:03,
[1/0]
B       172.27.1.0/24 [200/0] via 1.1.1.1 (recursive via 172.16.203.1, agg2), 00:37:30,
[1/0]
B       172.27.2.0/24 [200/0] via 1.1.1.1 (recursive via 172.16.203.1, agg2), 00:37:30,
[1/0]
B       172.27.5.0/24 [200/0] via 1.1.1.1 (recursive via 172.16.203.1, agg2), 00:37:30,
[1/0]
B       172.27.6.0/24 [200/0] via 1.1.1.1 (recursive via 172.16.203.1, agg2), 00:37:30,
[1/0]
B       172.27.7.0/24 [200/0] via 1.1.1.1 (recursive via 172.16.203.1, agg2), 00:37:30,
[1/0]
B       172.27.8.0/24 [200/0] via 1.1.1.1 (recursive via 172.16.203.1, agg2), 00:37:30,
[1/0]
B       172.29.1.0/24 [200/0] via 1.1.1.1 (recursive via 172.16.203.1, agg2), 00:37:30,
[1/0]
B       172.29.2.0/24 [200/0] via 1.1.1.1 (recursive via 172.16.203.1, agg2), 00:37:30,
[1/0]
```

To verify the BGP routing table for IPv6 and confirm the conditional advertisement occurred:

```
# get router info6 routing-table bgp
Routing table for VRF=0
B       2003:172:22:1::/64 [200/0] via 2003::1:1:1:1 (recursive via 2003:172:16:203::1,
agg2), 00:00:01, [1024/0]
B       2003:172:28:1::/64 [200/0] via 2003::3:3:3:3 (recursive via fe80::a5b:eff:feeb:ca45,
port1), 00:37:59, [1024/0]
B       2003:172:28:2::/64 [200/0] via 2003::3:3:3:3 (recursive via fe80::a5b:eff:feeb:ca45,
port1), 00:37:59, [1024/0]
```

Behavior when address family cross-checking is disabled

Using a similar BGP configuration with `cross-family-conditional-adv` disabled, note the following behavior based on the condition type.

When the condition type is set to exist:

```
config router bgp
       set cross-family-conditional-adv disable
       config neighbor
        edit "2.2.2.2"
            config conditional-advertise
                edit "222v4"
                    set condition-routemap "4-281" "6-281"
                    set condition-type exist
                next
            end
        next
    end
end
```

The FortiGate will only check the IPv4 RIB table to see if there is a matching IP address for each route map. Any IPv6 address under the route map will not get checked in the corresponding IPv6 RIB table, and the condition result will be non-existent. The `222v4` route map will not advertise to its neighbor because the result is non-existent, while the condition type is existent.

When the condition type is set to non-exist:

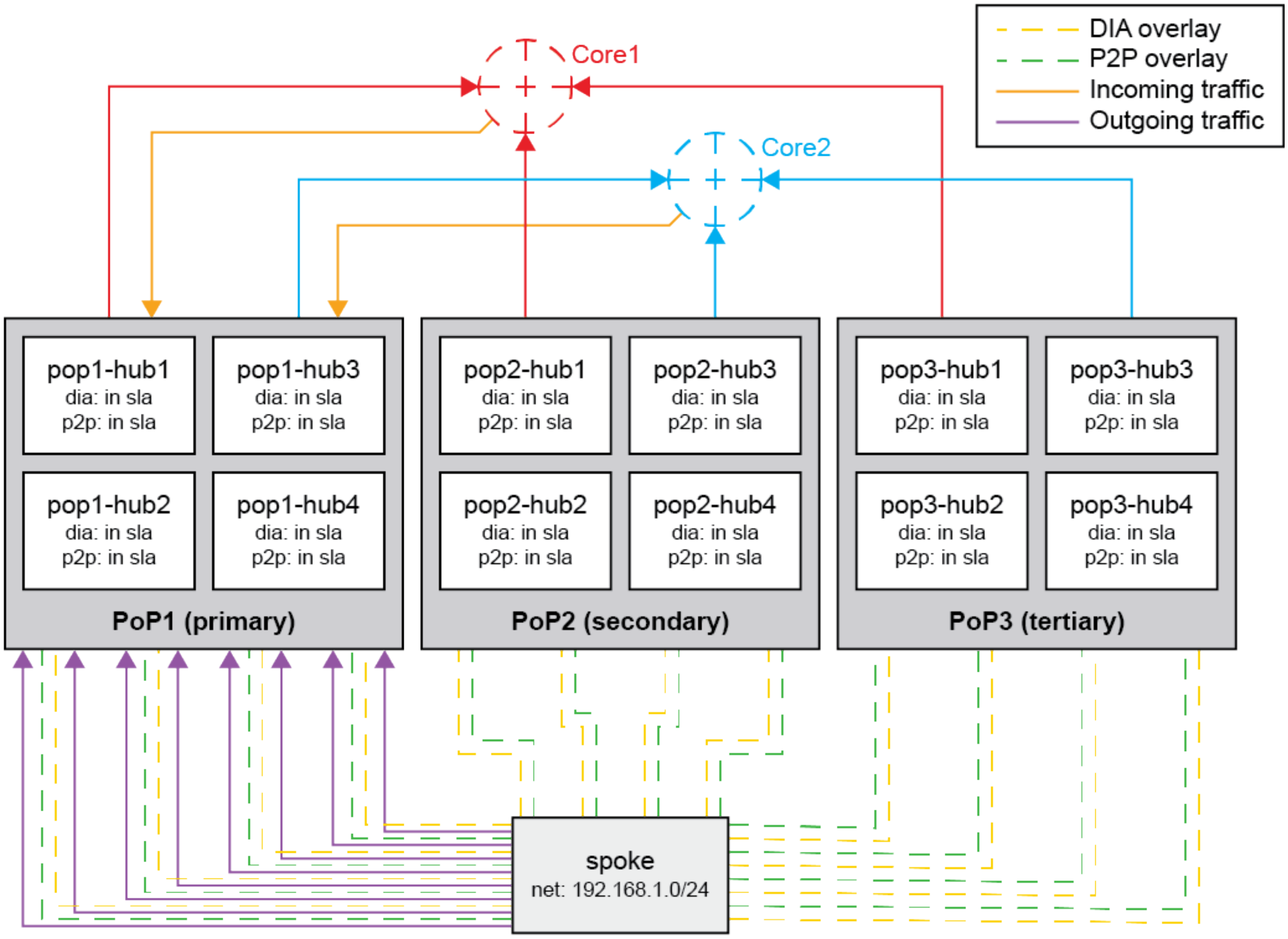
```
config router bgp
       set cross-family-conditional-adv disable
       config neighbor
        edit "2003::2:2:2:2"
            config conditional-advertise6
                edit "v6-222"
                    set condition-routemap "v6-238" "v4-287"
                    set condition-type non-exist
                next
            end
        next
    end
end
```

If the `v6-238` IPv6 prefix does not exist in the IPv6 RIB table, then the FortiGate will only check `v4-287` in the IPv6 RIB table. The FortiGate will not find it because it is an IPv4 address. Since the condition type is also `non-exist`, route `v6-222` will be advertised to its neighbor.

SD-WAN multi-PoP multi-hub large scale design and failover - 7.4.1

FortiOS 7.2.0 introduced a feature to define the minimum number of SD-WAN interface members that must meet SLA in order for the spoke to select a hub to process its SD-WAN traffic. This design is suitable for a single-PoP multi-hub architecture in order to achieve hub-to-hub failover. See Using multiple members per SD-WAN neighbor configuration for more information.

In FortiOS 7.4.1, the design is enhanced to support a multi-PoP multi-hub architecture in which incoming and outgoing traffic failover between PoPs is supported.



Based on the preceding diagram, incoming and outgoing traffic to the spoke is preferred over PoP1. If a single hub within PoP1 goes out of SLA, traffic will continue to flow through the PoP. If the minimum number of members to meet SLA in the PoP cannot be met, then traffic will fail over to PoP2.

The following enhancements have been made to support the multi-PoP failover scenario.

- Add `minimum-sla-meet-members` setting in the SD-WAN zone configurations and `zone-mode` setting in the SD-WAN service configurations:

```
config system sdwan
    config zone
        edit <name>
            set minimum-sla-meet-members <integer>
        next
    end
    config service
        edit <id>
            set mode sla
            set zone-mode {enable | disable}
        next
    end
end
```

When `zone-mode` is enabled on a SD-WAN service rule, the traffic is steered based on the status of the zone.

The state of the health check referenced in the SD-WAN service can be defined as follows:

  - If the number of in SLA members in a zone is less than the `minimum-sla-meet-members`, then the zone's state is out of SLA; otherwise, it is in SLA.
  - If a zone's state is out of SLA, then all members in the zone are out of SLA.
  - If a zone's state is in SLA, then the health check's state of individual members in the zone is determined by its own state.

- Add `service-id` setting in the SD-WAN neighbor configurations:

```
config system sdwan
    config neighbor
        edit <bgp_neighbor_ip>
            set member <member_id>
            set service-id <id>
        next
    end
end
```

The SD-WAN neighbor’s behavior can be determined by SD-WAN service and naturally synchronizes with SD-WAN service.

  - The SD-WAN service defines priority zones, whose SLA state determines the advertised community preferable string.
  - The SD-WAN service defines the `hold-down-time`, which determines how long an advertised community preferable string can be kept when it is expected to be changed.

- Add `sla-stickness` setting in the SD-WAN service configurations:

```
config system sdwan
    config service
        edit <id>
            set mode sla
            set sla-stickiness {enable | disable}
        next
    end
end
```

The switch-over of an existing session is determined as follows:

  - If the outgoing interface of the session is in SLA, then the session can keep its outgoing interface.
  - Otherwise, the session switches to a preferable path if one exists.

- Allow the neighbor group to be configured in the SD-WAN neighbor configurations:

```
config system sdwan
    config neighbor
        edit <bgp_neighbor_group>
            set member <member_id>
            set health-check <name>
            set sla-id <id>
        next
    end
end
```

Outgoing path control

The outgoing path from spoke to hub operates as follows:

1. Overlays to the primary and secondary PoP are assigned separately into an SD-WAN primary and secondary zone on the spoke.
2. One SD-WAN service rule is defined to include these zones as SD-WAN members.
3. When the primary zone is in SLA (`minimum-sla-meet-members` is met), the SD-WAN service rule steers traffic to the in SLA overlay members.
4. When the primary zone is out of SLA (`minimum-sla-meet-members` is not met), the SD-WAN service rule steers traffic to the in SLA overlay members in the secondary zone.
5. When the primary zone SLA is recovered:
   a. If `sla-stickness` is disabled on the SD-WAN service rule, then traffic will wait the duration of the `hold-down-time` before switching back to in SLA overlays in the primary zone.
   b. If `sla-stickness` is enabled on the SD-WAN service rule, then existing traffic will be kept on the in SLA overlays on the secondary zone, but new traffic will be steered to in SLA overlays in the primary zone.

Incoming path control

The incoming traffic from the core/external peers, to PoP, to spoke operates as follows:

1. When the primary zone is in SLA, the spoke uses the preferable route map to advertise local routes with the in SLA community to hubs in the primary and secondary PoPs.
   a. Hubs in the primary PoP translate the in SLA community into a short AS path and advertise it to external peers to attract incoming traffic.
   b. Hubs in the secondary PoP translate the in SLA community into a longer AS path and advertise it to external peers to deflect incoming traffic.
2. If the number of in SLA overlays in the primary zone is less than the `minimum-sla-meet-members`, then the spoke will use the default route map to advertise routes instead of with an out of SLA community to hubs in the primary PoP.
   a. Hubs in the primary PoP translate the out of SLA community into a longest AS path, and advertise it to external peers to deflect incoming traffic.
   b. As a result, inbound traffic is routed to hubs in the secondary PoP.
3. When the primary zone SLA is recovered:
   a. The spoke will wait the duration of the predefined `hold-down-time` in the SD-WAN service rule to use the preferable route map again to advertise routes with the in SLA community to hubs in the primary PoP.
   b. As a result, inbound traffic will be routed back to hubs in the primary PoP.

Neighbor group configuration

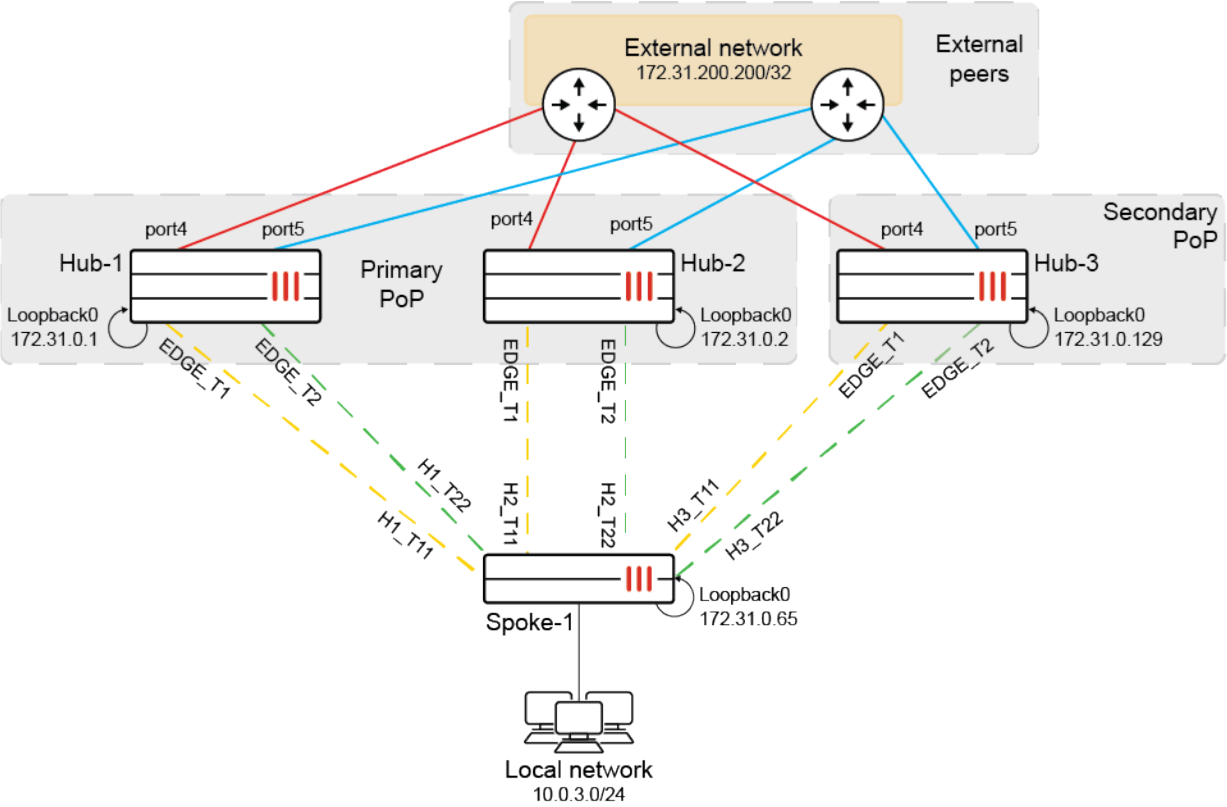
By configuring the neighbor group for spokes under the hub's SD-WAN neighbor configuration, if all paths from the hub to external peers are detected as out of SLA, then the hub will use the default route map to deny external routes to spokes that belong to this neighbor group defined on the hub. As a result, spokes will skip that specific hub and connect to external peers from other hubs.

This allows spokes to only measure overlay quality to each hub, and hubs to manage health checks to services by external peers. This significantly decreases the number of health check probes directly from the spoke to services and decreases the overall complexity. The complexity is further simplified by using multiple VRFs or segmentation where each spoke needs to send health check probes.

Example

This example configuration contains the following components:

- Two PoPs:
  - The primary PoP has two hubs (Hub-1 and Hub-2).
  - The secondary PoP has one hub (Hub-3).
- Spoke-1 has six overlays, with two overlay connections to each hub.
- Spoke-1 has three BGP neighbors, with one BGP neighbor for each hub.
  - All BGP neighbors are established on loopback IPs.
- Each hub has two paths to external peers.



Normally, outbound and inbound traffic go through hubs in the primary PoP. If the number of in SLA overlays to the primary PoP is less than the `minimum-sla-meet-members` (set to 2 in this example), bi-directional traffic needs to be switched to hubs in the secondary PoP. But when the primary PoP recovers and the `minimum-sla-meet-members` is met again, bi-directional traffic is forced back to hubs in the primary PoP after the predefined `hold-down-time` duration.

The hubs do not require SD-WAN configurations to the spokes. However, they use SD-WAN for connections to external peer routers.

Configuring the FortiGates

The following configurations highlight important routing and SD-WAN settings that must be configured on the spoke and the hubs. It is assumed that other configurations such as underlays, IPsec VPN overlays, loopbacks, static routes, and so on are already configured.

To configure Spoke-1:

1. Create the primary (PoP1) and secondary (PoP2) zones, and set the `minimum-sla-meet-members` to `2` on PoP1:

   ```
   config system sdwan
       set status enable
       config zone
           edit "virtual-wan-link"
           next
           edit "PoP1"
               set minimum-sla-meet-members 2
           next
           edit "PoP2"
           next
       end
   end
   ```

2. Add the overlay members to each zone. Four overlays are defined for PoP1, and two overlays are defined for PoP2:

   ```
   config system sdwan
       config members
           edit 1
               set interface "H1_T11"
               set zone "PoP1"
           next
           edit 2
               set interface "H1_T22"
               set zone "PoP1"
           next
           edit 3
               set interface "H2_T11"
               set zone "PoP1"
           next
           edit 4
               set interface "H2_T22"
               set zone "PoP1"
           next
           edit 5
               set interface "H3_T11"
               set zone "PoP2"
           next
           edit 6
               set interface "H3_T22"
               set zone "PoP2"
           next
       end
   end
   ```

3. Configure a performance SLA health check to a probe server behind the three hubs:

   ```
   config system sdwan
       config health-check
           edit "Hubs"
               set server "172.31.100.100"
   ```

```
            set source 172.31.0.65
            set members 0
            config sla
                edit 1
                    set link-cost-factor latency
                    set latency-threshold 200
                next
            end
        next
    end
end
```

4. Configure the service rule with the following settings: use SLA mode, enable zone mode to steer traffic based on the zone statuses, enable `sla-stickiness`, and use a 30-second hold down so that upon a recovery, existing sessions will remain on the secondary PoP while new sessions will switch back to the primary PoP once the 30-second duration ends:

```
config system sdwan
    config service
        edit 1
            set mode sla
            set zone-mode enable
            set dst "all"
            set src "CORP_LAN"
            set hold-down-time 30
            set sla-stickiness enable
            config sla
                edit "Hubs"
                    set id 1
                next
            end
            set priority-zone "PoP1" "PoP2"
        next
    end
end
```

   Since the PoP1 zone is specified before PoP2, PoP1 is regarded as the primary and preferred over the PoP2 zone.

5. Configure the in_sla and out_sla route maps that define the communities that are advertised to the hub when the zones are in and out of SLA.

   a. Configure the access list:

```
config router access-list
    edit "net10"
        config rule
            edit 1
                set prefix 10.0.3.0 255.255.255.0
            next
        end
    next
end
```

   b. Configure the route maps:

```
config router route-map
    edit "in_sla"
        config rule
            edit 1
```

```
                set match-ip-address "net10"
                set set-community "10:1"
            next
        end
    next
    edit "out_sla"
        config rule
            edit 1
                set match-ip-address "net10"
                set set-community "10:2"
            next
        end
    next
end
```

6. Configure the default route map for out of SLA scenarios, preferable route map for in SLA scenarios, and the local network to be advertised:

```
config router bgp
    config neighbor
        edit "172.31.0.1"
            ...
            set route-map-out "out_sla"
            set route-map-out-preferable "in_sla"
            ...
        next
        edit "172.31.0.2"
            ...
            set route-map-out "out_sla"
            set route-map-out-preferable "in_sla"
            ...
        next
        edit "172.31.0.129"
            ...
            set route-map-out "out_sla"
            set route-map-out-preferable "in_sla"
            ...
        next
    end
    config network
        edit 1
            set prefix 10.0.3.0 255.255.255.0
        next
    end
    ...
end
```

7. Define SD-WAN neighbors for each hub. The `minimum-sla-meet-members` is configured for the Hub-1 neighbor so that bi-directional traffic goes through Hub-1 as long as the in SLA overlays to Hub-1 are no less than 1. Associate the previously defined service rule to each SD-WAN neighbor:

```
config system sdwan
    config neighbor
        edit "172.31.0.1"
            set member 1 2
            set minimum-sla-meet-members 1
            set service-id 1
        next
        edit "172.31.0.2"
            set member 3 4
            set service-id 1
        next
        edit "172.31.0.129"
            set member 5 6
            set service-id 1
        next
    end
end
```

To configure the hubs:

1. Configure the SD-WAN zone, members, and health check for the external connections to peer routers. Performance SLA health checks are sent to external servers in order to measure the health of the external connections:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "port4"
        next
        edit 2
            set interface "port5"
        next
    end
    config health-check
        edit "external_peers"
            set server "10.0.1.2"
            set members 1 2
            config sla
                edit 1
                    set link-cost-factor latency
                    set latency-threshold 200
                next
            end
        next
    end
end
```

2. Configure the route maps for in and out of SLA scenarios. When out of SLA (one of the external connections is down), external routes are denied to be advertised to the spokes that are part of the neighbor group.
   a. Configure the access list:

```
config router access-list
    edit "net_Lo"
        config rule
            edit 1
                set prefix 172.31.200.200 255.255.255.255
            next
        end
    next
end
```

b. Configure the route maps:

```
    config router route-map
        edit "in_sla"
            config rule
                edit 1
                    set match-ip-address "net_Lo"
                next
            end
        next
        edit "out_sla"
            config rule
                edit 1
                    set action deny routes
                    set match-ip-address "net_Lo"
                next
            end
        next
    end
```

3. In the BGP settings, configure the external network prefix to advertise. Then configure the neighbor group and neighbor range for the spokes. Configure the preferable and default route maps to define the behavior when the external connections are in and out of SLA:

```
config router bgp
    ...
    config network
        edit 1
            set prefix 172.31.200.200 255.255.255.255
        next
    end
    config neighbor-group
        edit "EDGE"
            ...
            set route-map-out "out_sla"
            set route-map-out-preferable "in_sla"
            ...
        next
    end
    config neighbor-range
        edit 1
            set prefix 172.31.0.64 255.255.255.192
            set neighbor-group "EDGE"
        next
    end
    ...
end
```

4. Configure the SD-WAN neighbor to match the neighbor group that includes spokes as members. Specify that at least one of the external peer connections needs to be up to be considered in SLA:

```
config system sdwan
    config neighbor
```

```
            edit "EDGE"
                set member 1 2
                set minimum-sla-meet-members 1
                set health-check "external_peers"
                set sla-id 1
            next
        end
    end
```

Testing and verification

The following tests use diagnostic commands on various FortiGates to verify the connections in the SD-WAN configuration.

Test case 1: the primary PoP and Hub-1 are in SLA

To verify the configuration:

1. Verify the SD-WAN service rules status on Spoke-1. When all six overlays are in SLA on Spoke-1, the primary PoP and primary zone PoP1 are preferred. In particular, the overlay H1_T11 over PoP1 is preferred:

   ```
   Spoke-1 (root) # diagnose sys sdwan service
   Service(1): Address Mode(IPV4) flags=0x1c200 use-shortcut-sla use-shortcut sla-
   stickiness
    Tie break: cfg
     Gen(1), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-
   compare-order
   Hold down time(30) seconds, Hold start at 362646 second, now 362646
     Service role: standalone
     Members(6):
       1: Seq_num(1 H1_T11 PoP1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
   selected
       2: Seq_num(2 H1_T22 PoP1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
   selected
       3: Seq_num(3 H2_T11 PoP1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
   selected
       4: Seq_num(4 H2_T22 PoP1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
   selected
       5: Seq_num(5 H3_T11 PoP2), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
   selected
       6: Seq_num(6 H3_T22 PoP2), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
   selected
     Src address(1):
           10.0.0.0-10.255.255.255
     Dst address(1):
           0.0.0.0-255.255.255.255
   ```

2. Verify the BGP learned routes on Hub-1. The local route with in SLA community 10:1 is advertised to all hubs. Though, the AS paths on Hub-1 and Hub-2 are shorter than Hub-3:

   ```
   PoP1-Hub1 (root) # get router info bgp network 10.0.3.0/24
   VRF 0 BGP routing table entry for 10.0.3.0/24
   Paths: (1 available, best #1, table Default-IP-Routing-Table)
     Not advertised to any peer
     Original VRF 0
   ```

```
      Local, (Received from a RR-client)
        172.31.0.65 from 172.31.0.65 (172.31.0.65)
          Origin IGP metric 0, localpref 100, valid, internal, best
          Community: 10:1
          Last update: Mon Jul 17 15:16:57 2023
```

3. Send traffic from a host behind Spoke-1 to 172.31.200.200.
4. Run a sniffer trace on Spoke-1. Traffic leaves and returns on the H1_T11 overlay :

```
Spoke-1 (root) # diagnose sniffer packet any 'host 172.31.200.200' 4
interfaces=[any]
filters=[host 172.31.200.200]
5.098248 port4 in 10.0.3.2 -> 172.31.200.200: icmp: echo request
5.098339 H1_T11 out 10.0.3.2 -> 172.31.200.200: icmp: echo request
5.098618 H1_T11 in 172.31.200.200 -> 10.0.3.2: icmp: echo reply
5.098750 port4 out 172.31.200.200 -> 10.0.3.2: icmp: echo reply
```

Test case 2: a single SD-WAN member on Hub-1 is out of SLA

Hub-1 and PoP1 are still preferred in this scenario.

To verify the configuration:

1. Verify the health check status on Spoke-1. The H1_T11 overlay on Hub-1/PoP1 is out of SLA:

```
Spoke-1 (root) # diagnose sys sdwan health-check
Health Check(Hubs):
Seq(1 H1_T11): state(alive), packet-loss(0.000%) latency(220.214), jitter(0.015), mos
(4.104), bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x0
Seq(2 H1_T22): state(alive), packet-loss(0.000%) latency(0.196), jitter(0.014), mos
(4.404), bandwidth-up(999999), bandwidth-dw(999999), bandwidth-bi(1999998) sla_map=0x1
Seq(3 H2_T11): state(alive), packet-loss(0.000%) latency(0.173), jitter(0.008), mos
(4.404), bandwidth-up(999998), bandwidth-dw(999997), bandwidth-bi(1999995) sla_map=0x1
…
```

2. Verify the SD-WAN neighbor status. The SD-WAN neighbor still displays Hub-1’s zone status as pass/alive:

```
Spoke-1 (root) # diagnose sys sdwan neighbor
SD-WAN neighbor status: hold-down(disable), hold-down-time(0), hold_boot_time(0)
        Selected role(standalone) last_secondary_select_time/current_time in seconds
0/436439
Neighbor(172.31.0.1): member(1 2)role(standalone)
        Health-check(:0)  sla-pass selected alive
Neighbor(172.31.0.2): member(3 4)role(standalone)
        Health-check(:0)  sla-pass selected alive
Neighbor(172.31.0.129): member(5 6)role(standalone)
        Health-check(:0)  sla-pass selected alive
```

3. Verify the SD-WAN service rules status. Spoke-1 steers traffic to the H1_T22 overlay through Hub-1:

```
Spoke-1 (root) # diagnose sys sdwan service
Service(1): Address Mode(IPV4) flags=0x1c200 use-shortcut-sla use-shortcut sla-
stickiness
 Tie break: cfg
  Gen(2), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-
compare-order
Hold down time(30) seconds, Hold start at 364162 second, now 364162
```

```
  Service role: standalone
  Members(6):
    1: Seq_num(2 H1_T22 PoP1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    2: Seq_num(3 H2_T11 PoP1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    3: Seq_num(4 H2_T22 PoP1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    4: Seq_num(5 H3_T11 PoP2), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    5: Seq_num(6 H3_T22 PoP2), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    6: Seq_num(1 H1_T11 PoP1), alive, sla(0x0), gid(0), cfg_order(0), local cost(0),
selected
  Src address(1):
        10.0.0.0-10.255.255.255
  Dst address(1):
        0.0.0.0-255.255.255.255
```

4. Verify the BGP learned routes on Hub-1. The hubs continue to receive community 10:1 from the spoke and continue to route incoming traffic through Hub-1:

```
PoP1-Hub1 (root) # get router info bgp network 10.0.3.0/24
VRF 0 BGP routing table entry for 10.0.3.0/24
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  Original VRF 0
  Local, (Received from a RR-client)
    172.31.0.65 from 172.31.0.65 (172.31.0.65)
      Origin IGP metric 0, localpref 100, valid, internal, best
      Community: 10:1
      Last update: Mon Jul 17 15:16:57 2023
```

5. Send traffic from a host behind Spoke-1 to 172.31.200.200.
6. Run a sniffer trace on Spoke-1. Traffic leaves and returns on the H1_T22 overlay:

```
Spoke-1 (root) # diagnose sniffer packet any 'host 172.31.200.200' 4
interfaces=[any]
filters=[host 172.31.200.200]
25.299006 port4 in 10.0.3.2 -> 172.31.200.200: icmp: echo request
25.299080 H1_T22 out 10.0.3.2 -> 172.31.200.200: icmp: echo request
25.299323 H1_T22 in 172.31.200.200 -> 10.0.3.2: icmp: echo reply
25.299349 port4 out 172.31.200.200 -> 10.0.3.2: icmp: echo reply
```

Test case 3: both SD-WAN members on Hub-1 are out of SLA

Other in SLA overlays in zone PoP1 though Hub-2 are still preferred over PoP2 in this scenario.

To verify the configuration:

1. Verify the health check status on Spoke-1. Both H1_T11 and H1_T22 overlays on Hub-1/PoP1 are out of SLA:

```
Spoke-1 (root) # diagnose sys sdwan health-check
Health Check(Hubs):
Seq(1 H1_T11): state(alive), packet-loss(0.000%) latency(220.220), jitter(0.018), mos
(4.103), bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x0
Seq(2 H1_T22): state(alive), packet-loss(0.000%) latency(220.174), jitter(0.007), mos
```

```
(4.104), bandwidth-up(999999), bandwidth-dw(999999), bandwidth-bi(1999998) sla_map=0x0
Seq(3 H2_T11): state(alive), packet-loss(0.000%) latency(0.184), jitter(0.015), mos
(4.404), bandwidth-up(999998), bandwidth-dw(999997), bandwidth-bi(1999995) sla_map=0x1
Seq(4 H2_T22): state(alive), packet-loss(0.000%) latency(0.171), jitter(0.008), mos
(4.404), bandwidth-up(999999), bandwidth-dw(999999), bandwidth-bi(1999998) sla_map=0x1
Seq(5 H3_T11): state(alive), packet-loss(0.000%) latency(0.173), jitter(0.011), mos
(4.404), bandwidth-up(999999), bandwidth-dw(999999), bandwidth-bi(1999998) sla_map=0x1
Seq(6 H3_T22): state(alive), packet-loss(0.000%) latency(0.179), jitter(0.011), mos
(4.404), bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x1
```

2. Verify the SD-WAN neighbor status. The SD-WAN neighbor displays Hub-1's zone status as failed. However, SD-WAN Hub-2 is pass/alive:

```
Spoke-1 (root) # diagnose sys sdwan neighbor
SD-WAN neighbor status: hold-down(disable), hold-down-time(0), hold_boot_time(0)
        Selected role(standalone) last_secondary_select_time/current_time in seconds
0/436535
Neighbor(172.31.0.1): member(1 2)role(standalone)
        Health-check(:0)  sla-fail alive
Neighbor(172.31.0.2): member(3 4)role(standalone)
        Health-check(:0)  sla-pass selected alive
Neighbor(172.31.0.129): member(5 6)role(standalone)
        Health-check(:0)  sla-pass selected alive
```

3. Verify the SD-WAN service rules status. Spoke-1 steers traffic to the H2_T11 overlay through Hub-2:

```
Spoke-1 (root) # diagnose sys sdwan service
Service(1): Address Mode(IPV4) flags=0x1c200 use-shortcut-sla use-shortcut sla-
stickiness
 Tie break: cfg
  Gen(3), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-
compare-order
Hold down time(30) seconds, Hold start at 364489 second, now 364490
  Service role: standalone
  Members(6):
    1: Seq_num(3 H2_T11 PoP1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    2: Seq_num(4 H2_T22 PoP1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    3: Seq_num(5 H3_T11 PoP2), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    4: Seq_num(6 H3_T22 PoP2), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    5: Seq_num(1 H1_T11 PoP1), alive, sla(0x0), gid(0), cfg_order(0), local cost(0),
selected
    6: Seq_num(2 H1_T22 PoP1), alive, sla(0x0), gid(0), cfg_order(0), local cost(0),
selected
  Src address(1):
        10.0.0.0-10.255.255.255
  Dst address(1):
        0.0.0.0-255.255.255.255
```

4. Verify the BGP learned routes on Hub-1 and Hub-2. Hub-2 and Hub-3 continue to receive community 10:1 from Spoke-1, but Hub-1 receives the out of SLA community of 10:2.

a. On Hub-1:

```
PoP1-Hub1 (root) # get router info bgp network 10.0.3.0/24
VRF 0 BGP routing table entry for 10.0.3.0/24
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  Original VRF 0
  Local, (Received from a RR-client)
    172.31.0.65 from 172.31.0.65 (172.31.0.65)
      Origin IGP metric 0, localpref 100, valid, internal, best
      Community: 10:2
      Last update: Mon Jul 17 18:08:58 2023
```

b. On Hub-2:

```
PoP1-Hub2 (root) # get router info bgp network 10.0.3.0/24
VRF 0 BGP routing table entry for 10.0.3.0/24
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  Original VRF 0
  Local, (Received from a RR-client)
    172.31.0.65 from 172.31.0.65 (172.31.0.65)
      Origin IGP metric 0, localpref 100, valid, internal, best
      Community: 10:1
      Last update: Mon Jul 17 15:31:43 2023
```

5. Send traffic from a host behind Spoke-1 to 172.31.200.200.
6. Run a sniffer trace on Spoke-1. Traffic leaves and returns on the H2_T11 overlay:

```
Spoke-1 (root) # diagnose sniffer packet any 'host 172.31.200.200' 4
interfaces=[any]
filters=[host 172.31.200.200]
13.726009 port4 in 10.0.3.2 -> 172.31.200.200: icmp: echo request
13.726075 H2_T11 out 10.0.3.2 -> 172.31.200.200: icmp: echo request

13.726354 H2_T11 in 172.31.200.200 -> 10.0.3.2: icmp: echo reply
13.726382 port4 out 172.31.200.200 -> 10.0.3.2: icmp: echo reply
```

Test case 4: three SD-WAN members on PoP1 are out of SLA

The number of in SLA overlays in zone PoP1 is less than the `minimum-sla-meet-members` in zone PoP1. The SD-WAN service rule for Hub-2 is forcibly marked as `sla(0x0)` or out of SLA.

To verify the configuration:

1. Verify the health check status on Spoke-1. All three H1_T11, H1_T22, and H2_T11 overlays on PoP1 are out of SLA:

```
Spoke-1 (root) # diagnose sys sdwan health-check
Health Check(Hubs):
Seq(1 H1_T11): state(alive), packet-loss(0.000%) latency(220.219), jitter(0.019), mos
(4.103), bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x0
Seq(2 H1_T22): state(alive), packet-loss(0.000%) latency(220.184), jitter(0.008), mos
(4.104), bandwidth-up(999999), bandwidth-dw(999999), bandwidth-bi(1999998) sla_map=0x0
Seq(3 H2_T11): state(alive), packet-loss(0.000%) latency(220.171), jitter(0.009), mos
(4.104), bandwidth-up(999998), bandwidth-dw(999997), bandwidth-bi(1999995) sla_map=0x0
Seq(4 H2_T22): state(alive), packet-loss(0.000%) latency(0.180), jitter(0.013), mos
```

```
(4.404), bandwidth-up(999999), bandwidth-dw(999999), bandwidth-bi(1999998) sla_map=0x1
Seq(5 H3_T11): state(alive), packet-loss(0.000%) latency(0.174), jitter(0.014), mos
(4.404), bandwidth-up(999999), bandwidth-dw(999999), bandwidth-bi(1999998) sla_map=0x1
Seq(6 H3_T22): state(alive), packet-loss(0.000%) latency(0.179), jitter(0.015), mos
(4.404), bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x1
```

2. Verify the SD-WAN neighbor status. The SD-WAN neighbor displays Hub-1 and Hub-2’s zone status as failed:

```
Spoke-1 (root) # diagnose sys sdwan neighbor
SD-WAN neighbor status: hold-down(disable), hold-down-time(0), hold_boot_time(0)
        Selected role(standalone) last_secondary_select_time/current_time in seconds
0/436605
Neighbor(172.31.0.1): member(1 2)role(standalone)
        Health-check(:0)  sla-fail alive
Neighbor(172.31.0.2): member(3 4)role(standalone)
        Health-check(:0)  sla-fail alive
Neighbor(172.31.0.129): member(5 6)role(standalone)
        Health-check(:0)  sla-pass selected alive
```

3. Verify the SD-WAN service rules status. Since the minimum SLA members is not met for the primary zone (PoP1), the remaining overlay in PoP1 associated with the SD-WAN service rule is forcibly set to out of SLA. Spoke-1 steers traffic to the H3_T11 overlay through Hub-3:

```
Spoke-1 (root) # diagnose sys sdwan service
Service(1): Address Mode(IPV4) flags=0x1c200 use-shortcut-sla use-shortcut sla-
stickiness
 Tie break: cfg
  Gen(6), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-
compare-order
Hold down time(30) seconds, Hold start at 365341 second, now 365341
  Service role: standalone
  Members(6):
    1: Seq_num(5 H3_T11 PoP2), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    2: Seq_num(6 H3_T22 PoP2), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    3: Seq_num(1 H1_T11 PoP1), alive, sla(0x0), gid(0), cfg_order(0), local cost(0),
selected
    4: Seq_num(2 H1_T22 PoP1), alive, sla(0x0), gid(0), cfg_order(0), local cost(0),
selected
    5: Seq_num(3 H2_T11 PoP1), alive, sla(0x0), gid(0), cfg_order(0), local cost(0),
selected
    6: Seq_num(4 H2_T22 PoP1), alive, sla(0x0), gid(0), cfg_order(0), local cost(0),
selected
  Src address(1):
        10.0.0.0-10.255.255.255
  Dst address(1):
        0.0.0.0-255.255.255.255
```

4. Verify the BGP learned routes on each hub. Hub-3 continues to receive community 10:1 from Spoke-1, but Hub-1 and Hub-2 receive the out of SLA community of 10:2.

   a. On Hub-1:

   ```
   PoP1-Hub1 (root) # get router info bgp network 10.0.3.0/24
   VRF 0 BGP routing table entry for 10.0.3.0/24
   Paths: (1 available, best #1, table Default-IP-Routing-Table)
     Not advertised to any peer
   ```

```
    Original VRF 0
    Local, (Received from a RR-client)
      172.31.0.65 from 172.31.0.65 (172.31.0.65)
        Origin IGP metric 0, localpref 100, valid, internal, best
        Community: 10:2
        Last update: Mon Jul 17 18:22:14 2023
```

b. On Hub-2:

```
PoP1-Hub2 (root) # get router info bgp network 10.0.3.0/24
VRF 0 BGP routing table entry for 10.0.3.0/24
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  Original VRF 0
  Local, (Received from a RR-client)
    172.31.0.65 from 172.31.0.65 (172.31.0.65)
      Origin IGP metric 0, localpref 100, valid, internal, best
      Community: 10:2
      Last update: Mon Jul 17 18:37:53 2023
```

c. On Hub-3:

```
PoP2-Hub3 (root) # get router info bgp network 10.0.3.0/24
VRF 0 BGP routing table entry for 10.0.3.0/24
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  Original VRF 0
  Local, (Received from a RR-client)
    172.31.0.65 from 172.31.0.65 (172.31.0.65)
      Origin IGP metric 0, localpref 100, valid, internal, best
      Community: 10:1
      Last update: Mon Jul 17 14:39:04 2023
```

5. Send traffic from a host behind Spoke-1 to 172.31.200.200.
6. Run a sniffer trace on Spoke-1. Traffic leaves and returns on the H3_T11 overlay:

```
Spoke-1 (root) # diagnose sniffer packet any 'host 172.31.200.200' 4
interfaces=[any]
filters=[host 172.31.200.200]
38.501449 port4 in 10.0.3.2 -> 172.31.200.200: icmp: echo request
38.501519 H3_T11 out 10.0.3.2 -> 172.31.200.200: icmp: echo request
38.501818 H3_T11 in 172.31.200.200 -> 10.0.3.2: icmp: echo reply
38.501845 port4 out 172.31.200.200 -> 10.0.3.2: icmp: echo reply
```

Test case 5: an SD-WAN member on PoP1 recovers

SD-WAN member H2_T11 recovers and brings the number of overlays in SLA back to being above the `minimum-sla-meet-members` threshold in PoP1. After the hold down time duration (30 seconds), in SLA overlays in zone PoP1 are preferred over PoP2 again. With `sla-stickiness` enabled, existing traffic is kept on H3_T11, but new traffic is steered to H2_T11.

To verify the configuration:

1. Verify the SD-WAN service rules status on Spoke-1. The hold down timer has not yet passed, so H2_T11 is not yet preferred—even though the SLA status is pass/alive:

```
Spoke-1 (root) # diagnose sys sdwan service

Service(1): Address Mode(IPV4) flags=0x1c200 use-shortcut-sla use-shortcut sla-
stickiness
 Tie break: cfg
  Gen(16), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-
compare-order
Hold down time(30) seconds, Hold start at 431972 second, now 432000
  Service role: standalone
  Members(6):
    1: Seq_num(5 H3_T11 PoP2), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    2: Seq_num(6 H3_T22 PoP2), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    3: Seq_num(1 H1_T11 PoP1), alive, sla(0x0), gid(0), cfg_order(0), local cost(0),
selected
    4: Seq_num(2 H1_T22 PoP1), alive, sla(0x0), gid(0), cfg_order(0), local cost(0),
selected
    5: Seq_num(3 H2_T11 PoP1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    6: Seq_num(4 H2_T22 PoP1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
```

2. Verify the SD-WAN service rules status again after the hold down timer passes. H2_T11 and H2_T22 from PoP1 are now preferred:

```
Spoke-1 (root) # diagnose sys sdwan service
Service(1): Address Mode(IPV4) flags=0x1c200 use-shortcut-sla use-shortcut sla-
stickiness
 Tie break: cfg
  Gen(17), TOS(0x0/0x0), Protocol(0): src(1->65535):dst(1->65535), Mode(sla), sla-
compare-order
Hold down time(30) seconds, Hold start at 432003 second, now 432003
  Service role: standalone
  Members(6):
    1: Seq_num(3 H2_T11 PoP1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    2: Seq_num(4 H2_T22 PoP1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    3: Seq_num(5 H3_T11 PoP2), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    4: Seq_num(6 H3_T22 PoP2), alive, sla(0x1), gid(0), cfg_order(1), local cost(0),
selected
    5: Seq_num(1 H1_T11 PoP1), alive, sla(0x0), gid(0), cfg_order(0), local cost(0),
selected
    6: Seq_num(2 H1_T22 PoP1), alive, sla(0x0), gid(0), cfg_order(0), local cost(0),
selected
```

3. Verify the BGP learned routes on Hub-2, which now receives community 10:1 from Spoke-1:

```
PoP1-Hub2 (root) #  get router info bgp network 10.0.3.0/24
VRF 0 BGP routing table entry for 10.0.3.0/24
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  Original VRF 0
  Local, (Received from a RR-client)
    172.31.0.65 from 172.31.0.65 (172.31.0.65)
```

```
        Origin IGP metric 0, localpref 100, valid, internal, best
        Community: 10:1
        Last update: Tue Jul 18 14:41:32 2023
```

4. Send traffic from a host behind Spoke-1 to 172.31.200.200.
5. Run a sniffer trace on Spoke-1. Because of `sla-stickiness`, the existing traffic is kept on H3_T11:

```
Spoke-1 (root) # diagnose sniffer packet any 'host 172.31.200.200' 4
interfaces=[any]
filters=[host 172.31.200.200]

0.202708 port4 in 10.0.3.2 -> 172.31.200.200: icmp: echo request
0.202724 H3_T11 out 10.0.3.2 -> 172.31.200.200: icmp: echo request
0.202911 H3_T11 in 172.31.200.200 -> 10.0.3.2: icmp: echo reply
0.202934 port4 out 172.31.200.200 -> 10.0.3.2: icmp: echo reply
```

Test case 6: Hub-1 has an in SLA path to external peers

Since Hub-1 has an in SLA path to external peers, it will advertise the external route with destination 172.31.200.200/32 to Spoke-1.

To verify the configuration:

1. Verify the health check status on Hub-1. Note that port4 meets SLA, but port5 does not:

```
PoP1-Hub1 (root) # diagnose sys sdwan health-check
Health Check(external_peers):
Seq(1 port4): state(alive), packet-loss(0.000%) latency(0.161), jitter(0.009), mos
(4.404), bandwidth-up(999999), bandwidth-dw(999999), bandwidth-bi(1999998) sla_map=0x1
Seq(2 port5): state(dead), packet-loss(100.000%) sla_map=0x0
```

2. Verify the SD-WAN neighbor status. The `minimum-sla-meet-members` threshold of 1 is still met:

```
PoP1-Hub1 (root) # diagnose sys sdwan neighbor
Neighbor(EDGE): member(1 2)role(standalone)
        Health-check(external_peers:1)  sla-pass selected alive
```

3. Verify the BGP learned routes. Hub-1 still advertises the external route to the Spoke-1 BGP neighbor:

```
PoP1-Hub1 (root) # get router info bgp neighbors 172.31.0.65 advertised-routes
VRF 0 BGP table version is 13, local router ID is 172.31.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network                 Next Hop      Metric     LocPrf  Weight  RouteTag Path
*>i172.31.200.200/32     172.31.0.1     100        32768        0          i <-/->
Total number of prefixes 1
```

Test case 7: all external peers on Hub-1 are out of SLA

In this case, Hub-1 will now advertise the default route map, which denies the advertisement of the external route. Spoke-1 will now route traffic to the next hub.

To verify the configuration:

1. Verify the health check status on Hub-1. Note that port4 and port5 do not meet SLA:

   ```
   PoP1-Hub1 (root) # diagnose sys sdwan health-check
   Health Check(external_peers):
   Seq(1 port4): state(dead), packet-loss(100.000%) sla_map=0x0
   Seq(2 port5): state(dead), packet-loss(100.000%) sla_map=0x0
   ```

2. Verify the SD-WAN neighbor status. The `minimum-sla-meet-members` threshold of 1 is not met:

   ```
   PoP1-Hub1 (root) # diagnose sys sdwan neighbor
   Neighbor(EDGE): member(1 2)role(standalone)
           Health-check(external_peers:1)  sla-fail dead
   ```

3. Verify the BGP learned routes. Hub-1 does not advertise any external routes to the Spoke-1 BGP neighbor:

   ```
   PoP1-Hub1 (root) # get router info bgp neighbors 172.31.0.65 advertised-routes
   % No prefix for neighbor 172.31.0.65
   ```

Support IPsec tunnel to change names - 7.4.2

IPsec tunnels can be renamed. When you rename an IPsec tunnel, all references to the tunnel, such as routing and policies, are automatically updated to reflect the new name.

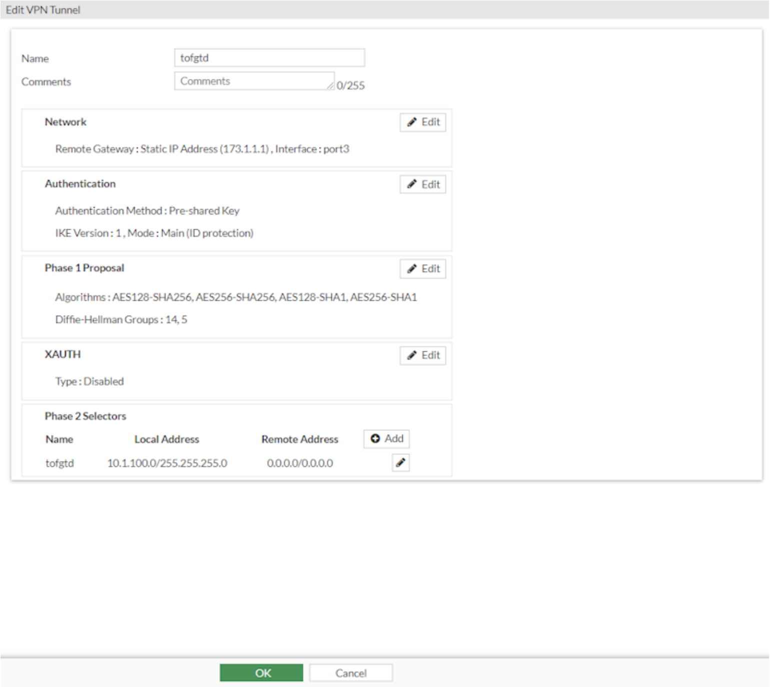
```
config vpn ipsec phase1-interface
    rename <string> to <string>
end
```

Example

In this example, the IPsec tunnel named *tofgtd* is renamed to *tofgtd-New*, and all associated references are updated.

To rename an IPsec tunnel in the GUI:

1. Go to *VPN > IPsec Tunnels* and double-click an IPsec tunnel to open it for editing.
   In this example, the IPsec tunnel name is *tofgtd*.



Name: tofgtd

Comments: Comments 0/255

Network

Remote Gateway : Static IP Address (173.1.1.1) , Interface : port3

Authentication

Authentication Method : Pre-shared Key

IKE Version : 1 , Mode : Main (ID protection)

Phase 1 Proposal

Algorithms : AES128-SHA256, AES256-SHA256, AES128-SHA1, AES256-SHA1

Diffie-Hellman Groups : 14, 5

XAUTH

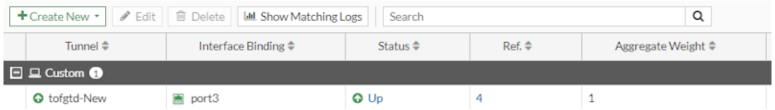
Type : Disabled

Phase 2 Selectors

Name	Local Address	Remote Address
tofgtd	10.1.100.0/255.255.255.0	0.0.0.0/0.0.0.0

2. In the *Name* box, type a new name, and click *OK*. The IPsec tunnel is renamed, and all associated references are updated.

   In this example, the IPsec tunnel is renamed to *tofgtd-New*.

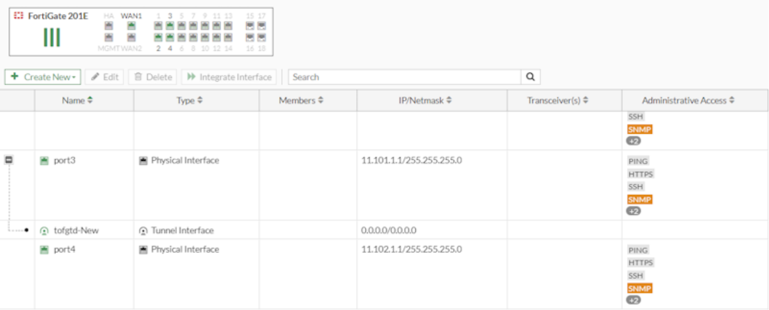


Tunnel	Interface Binding	Status	Ref.	Aggregate Weight
Custom 1				
tofgtd-New	port3	Up	4	1

3. Check the associated references:

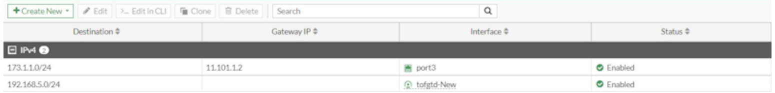
   In this example, all associated references show the new IPsec tunnel name of *tofgtd-New*.

   - Go to *Network > Interfaces* to see that the interface references the new IPsec tunnel name.



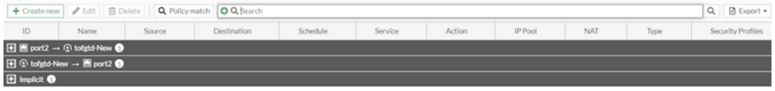
Name	Type	Members	IP/Netmask	Transceiver(s)	Administrative Access
					SSH SNMP +2
port3	Physical Interface		11.101.1.1/255.255.255.0		PING HTTPS SSH SNMP +2
tofgtd-New	Tunnel Interface		0.0.0.0/0.0.0.0		
port4	Physical Interface		11.102.1.1/255.255.255.0		PING HTTPS SSH SNMP +2

   - Go to *Network > Static Routes* to see that the static route references the new IPsec tunnel name.



Destination	Gateway IP	Interface	Status
IPv4 2			
173.1.1.0/24	11.101.1.2	port3	Enabled
192.168.5.0/24		tofgtd-New	Enabled

   - Go to *Policy & Objects > Firewall Policy* to see that the policy references the new IPsec tunnel name



ID	Name	Source	Destination	Schedule	Service	Action	IP Pool	NAT	Type	Security Profiles
port2 → tofgtd-New 1										
tofgtd-New → port2 1										
Implicit 1										

To rename an IPsec tunnel in the CLI:

1. Rename the IPsec tunnel.

   In this example, the IPsec tunnel named *tofgtd* is renamed to *tofgtd-New*:

   ```
   config vpn ipsec phase1-interface
       rename tofgtd to tofgtd-New
   end
   ```

2. Show the configuration to confirm that the IPsec tunnel was renamed.

   In this example, the IPsec tunnel was renamed to *tofgtd-New*:

   ```
   show
   config vpn ipsec phase1-interface
       edit "tofgtd-New"
           set interface "port3"
           set peertype any
           set net-device disable
           set proposal aes128-sha256 aes256-sha256 aes128-sha1 aes256-sha1
           set dpd disable
           set remote-gw 173.1.1.1
           ...
       next
   end
   ```

3. Check the associated references.

   In this example, all associated references show the new IPsec tunnel name of *tofgtd-New*.

   - Confirm that the interfaces reference the new IPsec tunnel name:

     ```
     config router static
     show
     config router static
         edit 3
             set dst 192.168.5.0 255.255.255.0
             set device "tofgtd-New"
         next
     end
     ```

   - Confirm that the static route references the new IPsec tunnel name:

     ```
     config system interface
     show
         edit "tofgtd-New"
         ....
     end
     ```

   - Confirm that the policies references the new IPsec tunnel name:

     ```
     config firewall policy
     show
     config firewall policy
         edit 1
             set uuid 802c6c2e-8368-51ee-bf40-6c3c32da1024
             set srcintf "port2"
             set dstintf "tofgtd-New"
             set action accept
             ...
         next
     ```

```
        edit 2
            set uuid 80d136aa-8368-51ee-cc52-b0b06306fb80
            set srcintf "tofgtd-New"
            set dstintf "port2"
            set action accept
            ...
        next
    end
```

Enhance IPv6 VRRP state control - 7.4.2

State control for IPv6 Virtual Router Redundancy Protocol (VRRP) is enhanced. Previously, the VRRP state would be *Primary* as long as any route, including the default route, could reach the IPv6 VRRP destination. Now administrators can choose whether to exclude the default route from the calculation of available routes to the IPv6 VRRP destination to better manage and control the VRRP states.

```
config system interface
    edit < name >
        config ipv6
        config vrrp6
            edit < id >
                set ignore-default-route {enable | disable}
            next
        end
    end
end
```

`set ignore-default-route {enable | disable}`	Set the default route to be ignored: • `enable`: Ignore the default route when checking the VRRP destination. • `disable`: Include the default route when checking the VRRP destination.

Example

In this example, the IPv6 VRRP destination (`vrdst6`) is set with an IPv6 address of `2000:172:22:20::22`, and `ignore-default-route` is enabled for the destination. As long as non-default routes exist to the VRRP destination, the VRRP state is `Primary`. When only the default route to the VRRP destination exists, the VRRP state changes to `Backup`.

To ignore the default route when checking the IPv6 VRRP destination:

1. Enable the default route to be ignored for IPv6 VRRP.

   In the following example, the IPv6 VRRP destination (`vrdst6`) is set with an IPv6 address of `2000:172:22:20::22`, and `ignore-default-route` is enabled for the destination.

   ```
   config system interface
       edit "port2"
           config ipv6
               set vrrp-virtual-mac6 enable
               set vrip6_link_local fe80::926c:acff:2222:2222
   ```

```
            config vrrp6
                edit 100
                    set vrgrp 100
                    set vrip6 2000:10:1:100::222
                    set priority 200
                    set vrdst6 2000:172:22:20::22
                    set ignore-default-route enable
                next
            end
        end
    next
end
```

2. Check the route for IPv6 VRRP destination.

   The following example, the routing table shows an active route through `port1` to the IPv6 VRRP destination of `2000:172:22:20::22`. The active route is not a default route.

   ```
   # get router info6 routing-table 2000:172:22:20::22
   Routing entry for 2000:172:22:20::/80
     Known via "static", distance 10, metric 0
     Last update 00:00:15 ago
       via 2000:172:16:200::55, port1
   ```

3. Check VRRP group information for IPv6.

   In the following example, the VRRP state is `Primary` because non-default routes to the IPv6 VRRP destination exist as shown in the previous step.

   ```
   # get router info6 vrrp
   Interface: port2, primary IPv6 address: 2000:10:1:100::1
   link-local IPv6 address: fe80::96f3:92ff:fe15:1ecd
   Virtual link-local IPv6 address: fe80::926c:acff:2222:2222
     UseVMAC: 1, SoftSW: 0, EmacVlan: 0 BrPortIdx: 0, PromiscCount: 1
     HA mode: primary (0:0:1)
     VRT primary count: 1
     VRID: 100 version: 3
       vrip: 2000:10:1:100::222, priority: 200, state: PRIMARY
       adv_interval: 1, preempt: 1, ignore_dft: 0, start_time: 3
       primary_adv_interval: 100, accept: 1
       vrmac: 00:00:5e:00:02:64
       vrdst: 2000:172:22:20::22
       vrgrp: 100
   ```

4. Delete the non-default routes to the IPv6 VRRP destination (`vrdst6`), and check the routes again.

   In the following example, the routing table shows only the default route (`::/0`) is available to the IPv6 VRRP destination of `2000:172:22:20::22`.

   ```
   # get router info6 routing-table 2000:172:22:20::22
   Routing entry for ::/0
     Known via "static", distance 10, metric 0, best
     Last update 02:02:09 ago
     * via 2000:172:16:200::254, port1
   ```

5. Check VRRP group information for IPv6.

   In the following example, the VRRP state is `Backup` because only the default route is available to the IPv6 VRRP destination as shown in the previous step.

```
#get router info6 vrrp
Interface: port2, primary IPv6 address: 2000:10:1:100::1
link-local IPv6 address: fe80::96f3:92ff:fe15:1ecd
Virtual link-local IPv6 address: fe80::926c:acff:2222:2222
  UseVMAC: 1, SoftSW: 0, EmacVlan: 0 BrPortIdx: 0, PromiscCount: 0
  HA mode: primary (0:0:1)
  VRT primary count: 0
  VRID: 100 version: 3
    vrip: 2000:10:1:100::222, priority: 0, state: BACKUP
    adv_interval: 1, preempt: 1, ignore_dft: 1, start_time: 3 but
    primary_adv_interval: 100, accept: 1
    vrmac: 00:00:5e:00:02:64
    vrdst: 2000:172:22:20::22
    vrgrp: 100
```

SD-WAN hub and spoke speed test improvements - 7.4.2

SD-WAN hub and spoke speed tests include the following improvements:

- Speed test servers can be deployed on a hub or a spoke. When deployed on a hub, speed tests can be initiated from spokes, even when a spoke is behind a NAT device.
- Tests can be in upload or download direction.
- Both TCP and UDP protocols are supported.
- An egress-shaping profile can be applied to local, remote, or both local and remote IPsec tunnels or no IPsec tunnels.
- Custom speed-test listening ports can be configured.

The test measures the speeds of the link to each spoke so that QoS can be applied on the hub to the dynamic IPsec overlay tunnels to each spoke. When the speed test is initiated from the spoke, the results are cached on the spoke, but sent to the hub to be applied to the egress traffic shaping profile assigned to the IPsec overlay tunnel interface and the respective tunnel. For more information about SD-WAN hub and spoke speed tests, see Running speed tests from the hub to the spokes in dial-up IPsec tunnels.

When a speed-test server is enabled, two speed test daemons are started and listen on different ports for different purposes:

- The controller speed test daemon listens on the IPsec overlay interfaces to assign an access token to each incoming speed test for authentication.
- The speed test daemon listens on the IPsec underlay interfaces to handle the speed tests.

Each incoming speed test request must present the obtained access token to prevent random, unauthorized requests. Otherwise, the connection is closed immediately. As such, speed test access must be enabled on both the underlay and the IPsec overlay tunnel interfaces on the hub.

```
config system interface
    edit <interface>
        set allowaccess speed-test [other access] ...
    next
end
```

If the IPsec tunnel has a configured `exchange-ip`, speed test access must also be configured on the associated interface, such as the loopback interface.

New commands are available to configure custom speed-test listening ports for the speed test server:

```
config system global
    set speedtestd-server-port <integer>
    set speedtestd-ctrl-port <integer>
end
```

`set speedtestd-server-port <integer>`	Specify a custom port number (1024 - 65535, default = 5201) for the speed test daemon. The port is used to perform the speed test.
`set speedtestd-ctrl-port <integer>`	Specify a custom port number (1024 - 65535. default = 5200) for the controller speed test daemon. The port is used to assign access tokens for authentication prior to performing the speed test.

The speed test client can be a hub or a spoke and must have `system speed-test-schedule` configured and the `dynamic-server` setting enabled.

On the speed test client, specify whether and how to apply the test results in a shaping profile. The shaping profile must be configured in the phase1 interface before it can be used with a speed test.

```
config system speed-test-schedule
    edit <interface>
        set server-port <integer>
        set ctrl-port <integer>
        set update-shaper {disable | local | remote | both}
    next
end
```

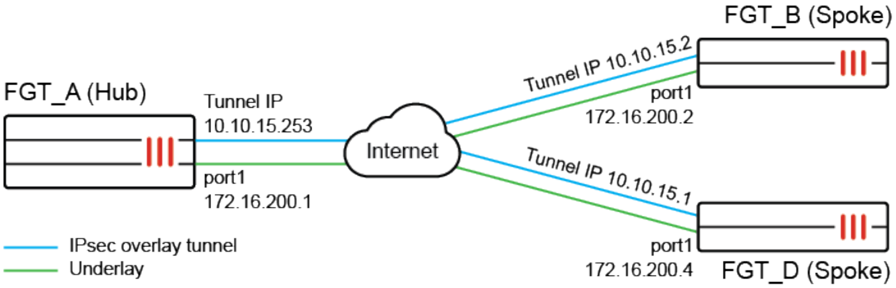
`set server-port <integer>`	Specify the port number for the speed-test server used for speed tests (1 - 65535, default = 5201).
`set ctrl-port <integer>`	Specify the port number for the controller on the speed-test server used for authentication (1 - 65535, default = 5200).
`set update-shaper {disable | local | remote | both}`	Set the egress shaper to use the speed test results: • `disable`: Disable updating the egress shaper (default). • `local`: Update the speed-test client egress shaper. • `remote`: Update the speed-test server egress shaper. • `both`: Update both the local and remote egress shapers.

Example

In this hub and spoke example, the hub is configured as an IPsec VPN dial-up server with two IPsec tunnels, and each tunnel is connected to a spoke. The VPN interfaces and IP addresses are:

FortiGate	Interface	IP Address
FGT_A (Hub)	hub-phase1	10.10.15.253
FGT_B (Spoke)	spoke11-p1	10.10.15.2
FGT_D (Spoke)	spoke21-p1	10.10.15.1

The hub (FGT_A) is configured as a speed-test server to listen on custom ports (6000 and 7000), and the spokes (FGT_B and FGT_D) are configured as speed-test clients. This setup allows speed tests to successfully perform when spokes are behind NAT devices. The results of the speed test will be applied to the hub-phase1 overlay tunnel(s) as specified by the speed-test clients.



The spokes are configured to initiate speed tests on a schedule on UDP. After the speed test completes, the results are sent to the hub, and the hub applies the results on its IPsec tunnels as egress traffic shaping. The results are also cached and can be used if an IPsec tunnel is disconnected and reconnected again.

This example focuses on the key settings required to enable a hub as the speed-test server and the spokes as speed-test clients that initiate the speed tests. For a complete example about running speed tests from the hub, see Running speed tests from the hub to the spokes in dial-up IPsec tunnels.

To configure the hub FortiGate (FGT_A):

1. Configure a shaping profile:
   In this example, the shaping profile is named profile_1.

```
config firewall shaping-profile
    edit "profile_1"
        set default-class-id 2
        config shaping-entries
            edit 1
                set class-id 2
                set priority low
                set guaranteed-bandwidth-percentage 10
                set maximum-bandwidth-percentage 10
            next
            edit 2
                set class-id 3
                set priority medium
                set guaranteed-bandwidth-percentage 30
                set maximum-bandwidth-percentage 40
            next
            edit 3
                set class-id 4
                set guaranteed-bandwidth-percentage 20
                set maximum-bandwidth-percentage 60
            next
        end
    end
end
```

Three classes are used in the profile for low, medium, and high priority traffic. Each class is assigned a guaranteed and maximum bandwidth as a percentage of the measured bandwidth from the speed test.

2. Configure a shaping policy to assign certain traffic as a class ID:

   In this example, all traffic destined to the dialup tunnels are assigned class 3.

```
config firewall shaping-policy
    edit 2
        set service "ALL"
        set schedule "always"
        set dstintf "hub-phase1" "hub2-phase1"
        set class-id 3
        set srcaddr "all"
        set dstaddr "all"
    next
end
```

3. Enable a speed test server with custom speed-test listening ports:

   A speed test server is enabled on the hub. Port 7000 will run speed tests, and port 6000 will be the controller used to issue access tokens for speed test authentication.

```
config system global
    ...
    set speedtest-server enable
    set speedtestd-ctrl-port 6000
    set speedtestd-server-port 7000
end
```

4. Allow the speed test on the underlay:

```
config system interface
    edit "port1"
        set ip 172.16.200.1 255.255.255.0
        set allowaccess ping https ssh snmp http telnet fgfm radius-acct probe-response 
fabric speed-test
        ...
    next
end
```

5. Allow the speed test on the overlay and use the shaping profile in the interface:

   In this example, speed tests are allowed on the overlay, and the shaping profile (profile_1) is used on the hub phase1 interface (port1).

```
config system interface
    edit "hub-phase1"
        set ip 10.10.15.253 255.255.255.255
        set allowaccess ping speed-test
        set egress-shaping-profile "profile_1"
        ...
        set interface "port1"
    next
end
```

To configure the first spoke FortiGate (FGT_B):

1. Configure `system speed-test-schedule`:

   The protocol mode is set to UDP. The custom controller port used for authentication is set to 6000, and the custom port used to run the speed tests is set to 7000. The shaping profile is set to remote.

   ```
   config system speed-test-schedule
       edit "spoke11-p1"
           set mode UDP
           set schedules "1"
           set dynamic-server enable
           set ctrl-port 6000
           set server-port 7000
           set update-shaper remote
       next
   end
   ```

2. Configure a recurring schedule for the speed tests:

   Schedule 1 is set to start at 08:37 every day of the week.

   ```
   config firewall schedule recurring
       edit "1"
           set start 08:37
           set day sunday monday tuesday wednesday thursday friday saturday
       next
   end
   ```

To configure the second spoke FortiGate (FGT_D):

1. Configure a speed test schedule:

   The protocol mode is set to UDP. The custom controller port used for authentication is set to 6000, and the custom port used to run the speed tests is set to 7000. The shaping profile is set to remote.

   ```
   config system speed-test-schedule
       edit "spoke21-p1"
           set mode UDP
           set schedules "1"
           set dynamic-server enable
           set ctrl-port 6000
           set server-port 7000
           set update-shaper remote
       next
   end
   ```

2. Configure a recurring schedule for the speed tests:

   Schedule 1 is set to start at 08:37 every day of the week.

   ```
   config firewall schedule recurring
       edit "1"
           set start 08:37
           set day sunday monday tuesday wednesday thursday friday saturday
   ```

```
        next
    end
```

To view the speed test results:

1. After the speed test schedule runs, view the result on spoke FGT_B:

   On spoke FGT_B, authentication succeeds through port 6000, and the test runs on port 7000. UDP mode is used, and the test is successful.

```
# diagnose debug application speedtest -1

......
fcron_speedtest_ipsec_request_init()-464: root: spoke11-p1(spoke11-p1) id=003900d5
fd=24, init request=0.0.0.0:0 -> 10.10.15.253:6000, test=172.16.200.2:0 ->
172.16.200.1:7000: succeed.
......
[speedtest(2181)] start uploading test.
[speedtest(2181)] Connecting to host 172.16.200.1, port 7000
[speedtest(2181)] [ 26] local 172.16.200.2 port 17553 connected to 172.16.200.1 port
7000
[speedtest(2181)] [ ID] Interval           Transfer     Bitrate         Total Datagrams
[speedtest(2181)] [ 26]   0.00-1.00   sec   150 MBytes  1.26 Gbits/sec  107570
[speedtest(2181)] [ 26]   1.00-2.00   sec   149 MBytes  1.25 Gbits/sec  107120
[speedtest(2181)] [ 26]   2.00-3.00   sec   149 MBytes  1.25 Gbits/sec  107030
[speedtest(2181)] [ 26]   3.00-4.00   sec   149 MBytes  1.25 Gbits/sec  107210
[speedtest(2181)] [ 26]   4.00-5.00   sec   149 MBytes  1.25 Gbits/sec  107260
[speedtest(2181)] [ ID] Interval           Transfer     Bitrate         Jitter
 Lost/Total Datagrams
[speedtest(2181)] [ 26]   0.00-5.00   sec   747 MBytes  1.25 Gbits/sec  0.000 ms
 0/536190 (0%)  sender
[speedtest(2181)] [ 26]   0.00-5.00   sec   271 MBytes   454 Mbits/sec  0.000 ms
 341627/535995 (64%)  receiver
[speedtest(2181)] client(sender): bytes_recv=283777280, bytes_sent=782837400, sender_
time=5.000, recver_time=5.000
[speedtest(2181)] client(sender): up_speed:  454 Mbits/sec
[speedtest(2181)]
[speedtest(2181)] speed test Done.
[speedtest(2181)] start downloading test.
[speedtest(2181)] Connecting to host 172.16.200.1, port 7000
[speedtest(2181)] Reverse mode, remote host 172.16.200.1 is sending
[speedtest(2181)] [ 26] local 172.16.200.2 port 7998 connected to 172.16.200.1 port 7000
[speedtest(2181)] [ ID] Interval           Transfer     Bitrate         Jitter
 Lost/Total Datagrams
[speedtest(2181)] [ 26]   0.00-1.00   sec  54.6 MBytes   458 Mbits/sec  0.007 ms
 70745/109978 (64%)
[speedtest(2181)] [ 26]   1.00-2.00   sec  54.8 MBytes   460 Mbits/sec  0.008 ms
 67547/106917 (63%)
[speedtest(2181)] [ 26]   2.00-3.00   sec  54.9 MBytes   460 Mbits/sec  0.010 ms
 67543/106940 (63%)
[speedtest(2181)] [ 26]   3.00-4.00   sec  54.8 MBytes   460 Mbits/sec  0.006 ms
 67636/107024 (63%)
[speedtest(2181)] [ 26]   4.00-5.00   sec  54.9 MBytes   460 Mbits/sec  0.004 ms
 67421/106842 (63%)
[speedtest(2181)] [ ID] Interval           Transfer     Bitrate         Jitter
 Lost/Total Datagrams
[speedtest(2181)] [ 26]   0.00-5.00   sec   750 MBytes  1.26 Gbits/sec  0.000 ms
```

```
 0/538540 (0%)  sender
[speedtest(2181)] [ 26]   0.00-5.00   sec   274 MBytes   460 Mbits/sec  0.004 ms
 340892/537701 (63%)  receiver
[speedtest(2181)] client(recver): bytes_recv=287341140, bytes_sent=786268400, sender_
time=5.000, recver_time=5.001
[speedtest(2181)] client(recver): down_speed:  460 Mbits/sec
[speedtest(2181)]
[speedtest(2181)] speed test Done.
fcron_speedtest_notify_func()-1275: Speed test pid=2181 done

fcron_speedtest_on_test_finish()-1211: Test 3900d5 for 'spoke11-p1' succeed with
up=454043, down=459694
fcron_speedtest_save_results()-1144: Write logs to disk: succ=1, fail=0
fcron_speedtest_sync_results()-1172: Sync cached results to secondary devices.
```

2. After the speed test schedule runs, view the result on the spoke FGT_D:

   On spoke FGT_D, authentication succeeds through port 6000, and the test runs on port 7000. UDP mode is used, and the test is successful.

```
# diagnose debug application speedtest -1

......
fcron_speedtest_ipsec_request_init()-464: root: spoke21-p1(spoke21-p1) id=00380011
fd=25, init request=0.0.0.0:0 -> 10.10.15.253:6000, test=172.16.200.4:0 ->
172.16.200.1:7000: succeed.
......
[speedtest(4309)] start uploading test.
[speedtest(4309)] Connecting to host 172.16.200.1, port 7000
[speedtest(4309)] [ 27] local 172.16.200.4 port 15349 connected to 172.16.200.1 port
7000
[speedtest(4309)] [ ID] Interval           Transfer     Bitrate         Total Datagrams
[speedtest(4309)] [ 27]   0.00-1.00   sec   148 MBytes  1.24 Gbits/sec  105940
[speedtest(4309)] [ 27]   1.00-2.00   sec   148 MBytes  1.24 Gbits/sec  105990
[speedtest(4309)] [ 27]   2.00-3.00   sec   147 MBytes  1.24 Gbits/sec  105860
[speedtest(4309)] [ 27]   3.00-4.00   sec   148 MBytes  1.24 Gbits/sec  105960
[speedtest(4309)] [ 27]   4.00-5.00   sec   148 MBytes  1.24 Gbits/sec  106090
[speedtest(4309)] [ ID] Interval           Transfer     Bitrate         Jitter
 Lost/Total Datagrams
[speedtest(4309)] [ 27]   0.00-5.00   sec   738 MBytes  1.24 Gbits/sec  0.000 ms
 0/529840 (0%)  sender
[speedtest(4309)] [ 27]   0.00-5.00   sec   271 MBytes   454 Mbits/sec  0.000 ms
 335130/529650 (63%)  receiver
[speedtest(4309)] client(sender): bytes_recv=283999200, bytes_sent=773566400, sender_
time=5.000, recver_time=5.000
[speedtest(4309)] client(sender): up_speed:  454 Mbits/sec
[speedtest(4309)]
[speedtest(4309)] speed test Done.
[speedtest(4309)] start downloading test.
[speedtest(4309)] Connecting to host 172.16.200.1, port 7000
[speedtest(4309)] Reverse mode, remote host 172.16.200.1 is sending
[speedtest(4309)] [ 27] local 172.16.200.4 port 19586 connected to 172.16.200.1 port
7000
[speedtest(4309)] [ ID] Interval           Transfer     Bitrate         Jitter
 Lost/Total Datagrams
[speedtest(4309)] [ 27]   0.00-1.00   sec  56.1 MBytes   471 Mbits/sec  0.005 ms
 70258/110574 (64%)
```

```
[speedtest(4309)] [ 27]   1.00-2.00   sec  56.0 MBytes   470 Mbits/sec  0.006 ms
 66496/106740 (62%)
[speedtest(4309)] [ 27]   2.00-3.00   sec  56.0 MBytes   470 Mbits/sec  0.005 ms
 66481/106736 (62%)
[speedtest(4309)] [ 27]   3.00-4.00   sec  56.1 MBytes   471 Mbits/sec  0.007 ms
 66403/106690 (62%)
[speedtest(4309)] [ 27]   4.00-5.00   sec  56.3 MBytes   473 Mbits/sec  0.008 ms
 65991/106454 (62%)
[speedtest(4309)] [ ID] Interval           Transfer     Bitrate         Jitter
 Lost/Total Datagrams
[speedtest(4309)] [ 27]   0.00-5.00   sec   749 MBytes  1.26 Gbits/sec  0.000 ms
 0/538110 (0%)  sender
[speedtest(4309)] [ 27]   0.00-5.00   sec   281 MBytes   471 Mbits/sec  0.008 ms
 335629/537194 (62%)  receiver
[speedtest(4309)] client(recver): bytes_recv=294284900, bytes_sent=785640600, sender_
time=5.000, recver_time=5.001
[speedtest(4309)] client(recver): down_speed:  471 Mbits/sec
[speedtest(4309)]
[speedtest(4309)] speed test Done.
fcron_speedtest_notify_func()-1275: Speed test pid=4309 done

fcron_speedtest_on_test_finish()-1211: Test 380011 for 'spoke21-p1' succeed with
up=454398, down=470794
fcron_speedtest_save_results()-1144: Write logs to disk: succ=1, fail=0
fcron_speedtest_sync_results()-1172: Sync cached results to secondary devices.
```

3. After the speed test schedule runs, view the result on the hub (FGT_A):

   > The server side uses `speedtestd`, while the client side uses `speedtest`.

   The speed test results are applied on hub-phase1_0 and hub_phase1_1 as egress traffic shaping.

```
# diagnose debug application speedtestd -1

......
[speedtest(2771)] [  7] local 172.16.200.1 port 7000 connected to 172.16.200.2 port
17553
......
[speedtest(2771)] [  7] local 172.16.200.1 port 7000 connected to 172.16.200.2 port 7998
......
[sptestd::ctrl(0377):root] set shaper: if=hub-phase1, tun=hub-phase1_0, sp=profile_1,
bw=459745
......
[speedtest(2771)] [  7] local 172.16.200.1 port 7000 connected to 172.16.200.4 port
15349
......
[speedtest(2771)] [  7] local 172.16.200.1 port 7000 connected to 172.16.200.4 port
19586
......
[sptestd::ctrl(0377):root] set shaper: if=hub-phase1, tun=hub-phase1_1, sp=profile_1,
bw=470855
......
```

4. Verify the result is cached on the spokes.

- On FGT_B, the speed test results are cached:

  ```
  #  diagnose test application forticron 10
  Speed test results:
  1: vdom=root, phase1intf=spoke11-p1, peer-id='172.16.200.1', up=454043, dw=459694,
  time=12/13 12:32:19
  ```

- On FGT_D, the speed test results are cached:

  ```
  #   diagnose test application forticron 10
  Speed test results:
  1: vdom=root, phase1intf=spoke21-p1, peer-id='172.16.200.1', up=454398, dw=470794,
  time=12/12 16:33:18
  ```

5. On the hub (FGT_A), verify the speed test results are applied to the hub's IPsec tunnels as egress traffic shaping: On hub-phase1_0 and hub-phase1_1, the correct traffic control is displayed.

```
# diagnose vpn tunnel list
list all ipsec tunnel in vd 0
......
------------------------------------------------------
name=hub-phase1_0 ver=2 serial=16 172.16.200.1:0->172.16.200.2:0 tun_id=10.10.15.1 tun_
id6=2000:10:10:15::1 dst_mtu=1500 dpd-link=on weight=1
bound_if=11 lgwy=static/1 tun=intf mode=dial_inst/3 encap=none/74408 options[122a8]=npu
rgwy-chg frag-rfc  run_state=0 role=primary accept_traffic=1 overlay_id=10

parent=hub-phase1 index=0
......
egress traffic control:
        bandwidth=459745(kbps) lock_hit=0 default_class=2 n_active_class=3
        class-id=2      allocated-bandwidth=45974(kbps)         guaranteed-
bandwidth=45974(kbps)
                        max-bandwidth=45974(kbps)       current-bandwidth=0(kbps)
                        priority=low    forwarded_bytes=86K
                        dropped_packets=0       dropped_bytes=0
        class-id=3      allocated-bandwidth=137923(kbps)        guaranteed-
bandwidth=137923(kbps)
                        max-bandwidth=183897(kbps)      current-bandwidth=0(kbps)
                        priority=medium         forwarded_bytes=0
                        dropped_packets=0       dropped_bytes=0
        class-id=4      allocated-bandwidth=275846(kbps)        guaranteed-
bandwidth=91948(kbps)
                        max-bandwidth=275846(kbps)      current-bandwidth=0(kbps)
                        priority=high   forwarded_bytes=0
                        dropped_packets=0       dropped_bytes=0
------------------------------------------------------
name=hub-phase1_1 ver=2 serial=17 172.16.200.1:0->172.16.200.4:0 tun_id=10.10.15.2 tun_
id6=2000:10:10:15::2 dst_mtu=1500 dpd-link=on weight=1
bound_if=11 lgwy=static/1 tun=intf mode=dial_inst/3 encap=none/74408 options[122a8]=npu
rgwy-chg frag-rfc  run_state=0 role=primary accept_traffic=1 overlay_id=10

parent=hub-phase1 index=1
......
egress traffic control:
        bandwidth=470855(kbps) lock_hit=0 default_class=2 n_active_class=3
        class-id=2      allocated-bandwidth=47085(kbps)         guaranteed-
bandwidth=47085(kbps)
                        max-bandwidth=47085(kbps)       current-bandwidth=0(kbps)
```

```
                        priority=low    forwarded_bytes=81K
                        dropped_packets=0       dropped_bytes=0
        class-id=3      allocated-bandwidth=141256(kbps)        guaranteed-
bandwidth=141256(kbps)
                        max-bandwidth=188341(kbps)      current-bandwidth=0(kbps)
                        priority=medium         forwarded_bytes=0
                        dropped_packets=0       dropped_bytes=0
        class-id=4      allocated-bandwidth=282512(kbps)        guaranteed-
bandwidth=94170(kbps)
                        max-bandwidth=282512(kbps)      current-bandwidth=0(kbps)
                        priority=high   forwarded_bytes=0
                        dropped_packets=0       dropped_bytes=0
```

SD-WAN steering

7.4.0

7.4.1

7.4.2

Classifying SLA probes for traffic prioritization

This information is also available in the FortiOS 7.4 Administration Guide:

- Classifying SLA probes for traffic prioritization

Support for traffic classification on SLA probes has been implemented to ensure they are prioritized in times of congestion. This prevents SD-WAN link flapping and unexpected routing behaviors, and stabilizes SD-WAN from unnecessary failovers.

SLA probes can now be classified into a specific class ID so that SLA probes assigned to a class ID with higher priority are prioritized over other traffic. SLA probes are assigned using the `class-id` command:

```
config system sdwan
    config health-check
        edit <health-check name>
            set class-id <class name>
        next
    end
end
```

Example

In this example, SLA probes are assigned into different class ID. The interfaces dmz and vd1-01 both have outbandwidth of 1000000 Kbps (1 Gbps) configured. Three traffic shaping classes are defined:

Class ID	Name	Definition
2	sla_probe	High priority with a guaranteed 10% of bandwidth (100 Mbps)
3	default	Low priority with a guaranteed 80% of bandwidth (800 Mbps)
4	sla_probe_2	Medium priority with a guaranteed 10% of bandwidth (100 Mbps)

Under this scheme, when congestion occurs, traffic in each class will have their guaranteed bandwidth honored. If there is remaining bandwidth, higher priority traffic will get the bandwidth. On the SD-WAN health check, probes to server 2.2.2.2 are assigned to class 2 (sla_probe). This means it has a guaranteed bandwidth and has the highest priority to use unused bandwidth. This allows SD-WAN health check to function properly even during times of congestion.

To classify SLA probes for traffic prioritization:

1. Configure the firewall traffic class:

```
config firewall traffic-class
    edit 2
        set class-name "sla_probe"
    next
    edit 3
        set class-name "default"
    next
    edit 4
        set class-name "sla_probe_2"
    next
end
```

2. Configure the class ID priority and guaranteed bandwidth:

```
config firewall shaping-profile
    edit "profile-1"
        set default-class-id 3
        config shaping-entries
            edit 2
                set class-id 2
                set priority high
                set guaranteed-bandwidth-percentage 10
                set maximum-bandwidth-percentage 100
            next
            edit 3
                set class-id 3
                set priority low
                set guaranteed-bandwidth-percentage 80
                set maximum-bandwidth-percentage 100
            next
            edit 4
                set class-id 4
                set priority medium
                set guaranteed-bandwidth-percentage 10
                set maximum-bandwidth-percentage 100
            next
        end
    next
end
```

3. Configure the interfaces:

```
config system interface
    edit "dmz"
        set outbandwidth 1000000
        set egress-shaping-profile "profile-1"
        ...
    next
    edit "vd1-p1"
        set outbandwidth 1000000
        set egress-shaping-profile "profile-1"
        ...
    next
end
```

4. Configure the SD-WAN health check and assign the SLA probes into class 2:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "dmz"
            set gateway 172.16.208.2
        next
        edit 2
            set interface "vd1-p1"
        next
    end
    config health-check
        edit "1"
            set server "2.2.2.2"
            set members 1 2
            set class-id 2
            config sla
                edit 1
                next
            end
        next
    end
end
```

To verify the SLA probe assignment:

1. Verify the health check diagnostics:

```
diagnose sys sdwan health-check
    Health Check(1):
    Seq(1 dmz): state(alive), packet-loss(0.000%) latency(0.247), jitter(0.022), mos
(4.404), bandwidth-up(999999), bandwidth-dw(999997), bandwidth-bi(1999996) sla_map=0x1
    Seq(2 vd1-p1): state(alive), packet-loss(0.000%) latency(0.247), jitter(0.018), mos
(4.404), bandwidth-up(999999), bandwidth-dw(1000000), bandwidth-bi(1999999) sla_map=0x1
```

2. Verify the SLA probes are assigned into class 2:

```
# diagnose netlink interface list dmz
    if=dmz family=00 type=1 index=5 mtu=1500 link=0 master=0
    ref=36 state=start present fw_flags=10018000 flags=up broadcast run multicast
    Qdisc=mq hw_addr=e0:23:ff:9d:f9:9e broadcast_addr=ff:ff:ff:ff:ff:ff
    egress traffic control:
            bandwidth=1000000(kbps) lock_hit=0 default_class=3 n_active_class=3
            class-id=3      allocated-bandwidth=800000(kbps)        guaranteed-
bandwidth=800000(kbps)
                            max-bandwidth=1000000(kbps)     current-bandwidth=1(kbps)
                            priority=low    forwarded_bytes=1446
                            dropped_packets=0       dropped_bytes=0
            class-id=4      allocated-bandwidth=100000(kbps)        guaranteed-
bandwidth=100000(kbps)
                            max-bandwidth=1000000(kbps)     current-bandwidth=0(kbps)
                            priority=medium         forwarded_bytes=0
                            dropped_packets=0       dropped_bytes=0
            class-id=2      allocated-bandwidth=100000(kbps)        guaranteed-
bandwidth=100000(kbps)
                            max-bandwidth=1000000(kbps)     current-bandwidth=1(kbps)
                            priority=high   forwarded_bytes=1404
                            dropped_packets=0       dropped_bytes=0
    stat: rxp=19502 txp=14844 rxb=2233923 txb=802522 rxe=0 txe=0 rxd=0 txd=0 mc=0
collision=0 @ time=1675121675
    re: rxl=0 rxo=0 rxc=0 rxf=0 rxfi=0 rxm=0
    te: txa=0 txc=0 txfi=0 txh=0 txw=0
    misc rxc=0 txc=0
    input_type=0 state=3 arp_entry=0 refcnt=36
# diagnose netlink interface list vd1-p1
    if=vd1-p1 family=00 type=768 index=99 mtu=1420 link=0 master=0
    ref=20 state=start present fw_flags=10010000 flags=up p2p run noarp multicast
    Qdisc=noqueue
    egress traffic control:
            bandwidth=1000000(kbps) lock_hit=0 default_class=3 n_active_class=3
            class-id=3      allocated-bandwidth=800000(kbps)        guaranteed-
bandwidth=800000(kbps)
                            max-bandwidth=1000000(kbps)     current-bandwidth=0(kbps)
                            priority=low    forwarded_bytes=0
                            dropped_packets=0       dropped_bytes=0
            class-id=4      allocated-bandwidth=100000(kbps)        guaranteed-
bandwidth=100000(kbps)
                            max-bandwidth=1000000(kbps)     current-bandwidth=0(kbps)
                            priority=medium         forwarded_bytes=0
                            dropped_packets=0       dropped_bytes=0
            class-id=2      allocated-bandwidth=100000(kbps)        guaranteed-
```

```
bandwidth=100000(kbps)
                            max-bandwidth=1000000(kbps)     current-bandwidth=1(kbps)
                            priority=high   forwarded_bytes=1120
                            dropped_packets=0       dropped_bytes=0
    stat: rxp=4097 txp=4586 rxb=540622 txb=221500 rxe=0 txe=19 rxd=0 txd=0 mc=0
collision=0 @ time=1675121742
    re: rxl=0 rxo=0 rxc=0 rxf=0 rxfi=0 rxm=0
    te: txa=0 txc=0 txfi=0 txh=0 txw=0
    misc rxc=0 txc=0
    input_type=0 state=3 arp_entry=0 refcnt=20
```

When verifying the class assignment, the counter value should increase.

The example also demonstrates assigning SLA probes to class 4 (sla_probe_2), in which case the probes get medium priority.

To assign the SLA probe to medium priority:

1. Assign SLA probes into class 4:

```
config sys sdwan
    config health-check
        edit 1
            set class-id 4
        next
    end
    set status disable
end
config sys sdwan
    set status enable
end
```

2. Verify the SLA probes are assigned into class 4.

```
# diagnose netlink interface list dmz
    if=dmz family=00 type=1 index=5 mtu=1500 link=0 master=0
    ref=34 state=start present fw_flags=10018000 flags=up broadcast run multicast
    Qdisc=mq hw_addr=e0:23:ff:9d:f9:9e broadcast_addr=ff:ff:ff:ff:ff:ff
    egress traffic control:
            bandwidth=1000000(kbps) lock_hit=0 default_class=3 n_active_class=3
            class-id=3      allocated-bandwidth=800000(kbps)        guaranteed-
bandwidth=800000(kbps)
                            max-bandwidth=1000000(kbps)     current-bandwidth=1(kbps)
                            priority=low    forwarded_bytes=24K
                            dropped_packets=0       dropped_bytes=0
            class-id=4      allocated-bandwidth=100000(kbps)        guaranteed-
bandwidth=100000(kbps)
                            max-bandwidth=1000000(kbps)     current-bandwidth=1(kbps)
                            priority=medium         forwarded_bytes=1674
                            dropped_packets=0       dropped_bytes=0
            class-id=2      allocated-bandwidth=100000(kbps)        guaranteed-
bandwidth=100000(kbps)
                            max-bandwidth=1000000(kbps)     current-bandwidth=0(kbps)
                            priority=high   forwarded_bytes=0
                            dropped_packets=0       dropped_bytes=0
    stat: rxp=20818 txp=15874 rxb=2382789 txb=857674 rxe=0 txe=0 rxd=0 txd=0 mc=0
collision=0 @ time=1675122057
    re: rxl=0 rxo=0 rxc=0 rxf=0 rxfi=0 rxm=0
    te: txa=0 txc=0 txfi=0 txh=0 txw=0
    misc rxc=0 txc=0
    input_type=0 state=3 arp_entry=0 refcnt=34
# diagnose netlink interface list vd1-p1
    if=vd1-p1 family=00 type=768 index=99 mtu=1420 link=0 master=0
    ref=20 state=start present fw_flags=10010000 flags=up p2p run noarp multicast
    Qdisc=noqueue
    egress traffic control:
            bandwidth=1000000(kbps) lock_hit=0 default_class=3 n_active_class=3
            class-id=3      allocated-bandwidth=800000(kbps)        guaranteed-
bandwidth=800000(kbps)
                            max-bandwidth=1000000(kbps)     current-bandwidth=0(kbps)
                            priority=low    forwarded_bytes=0
                            dropped_packets=0       dropped_bytes=0
            class-id=4      allocated-bandwidth=100000(kbps)        guaranteed-
bandwidth=100000(kbps)
                            max-bandwidth=1000000(kbps)     current-bandwidth=1(kbps)
                            priority=medium         forwarded_bytes=1280
                            dropped_packets=0       dropped_bytes=0
            class-id=2      allocated-bandwidth=100000(kbps)        guaranteed-
bandwidth=100000(kbps)
                            max-bandwidth=1000000(kbps)     current-bandwidth=0(kbps)
                            priority=high   forwarded_bytes=0
                            dropped_packets=0       dropped_bytes=0
    stat: rxp=4097 txp=4703 rxb=540622 txb=226180 rxe=0 txe=19 rxd=0 txd=0 mc=0
collision=0 @ time=1675122058
    re: rxl=0 rxo=0 rxc=0 rxf=0 rxfi=0 rxm=0
    te: txa=0 txc=0 txfi=0 txh=0 txw=0
    misc rxc=0 txc=0
    input_type=0 state=3 arp_entry=0 refcnt=20
```

Support IPv6 application based steering in SD-WAN

This information is also available in the FortiOS 7.4 Administration Guide:

- Internet service and application control steering

IPv6 based SD-WAN rules allow matching of applications and application categories. The following options are available with `set addr-mode ipv6`:

```
config system sdwan
   config service
     edit
       set addr-mode ipv6
       set internet-service enable
       set internet-service-app-ctrl
       set internet-service-app-ctrl-group
       set internet-service-app-ctrl-category
```

```
        next
    end
end
```

Example

In this example, SD-WAN is configured to use an IPv6 service rule to steer traffic from FGT_A to FGT_B based on the following application control options:

- Application Telnet
- An application group for ping
- An application category that includes SSH

When the rule is matched, traffic is steered based on the lowest cost SLA strategy. In this example, vlan100 is the preferred interface, and traffic is routed to vlan100 on FGT_B.

To view the configuration:

1. View the SD-WAN configuration on FGT_A:

   SD-WAN has four members in the default virtual-wan-link zone, each with an IPv4 and IPv6 gateway. The SD-WAN service rule includes `internet-service-app-ctrl 16091` for the Telnet, `internet-service-app-ctrl-group "network-Ping"` for ping , and `internet-service-app-ctrl-category 15` for SSH applications.

   ```
   (sdwan) # show
   config system sdwan
       set status enable
       config zone
           edit "virtual-wan-link"
           next
       end
       config members
           edit 1
               set interface "dmz"
               set gateway 172.16.208.2
               set gateway6 2000:172:16:208::2
           next
           edit 2
               set interface "IPSec-1"
           next
           edit 3
               set interface "agg1"
               set gateway 172.16.203.2
               set gateway6 2000:172:16:203::2
           next
           edit 4
               set interface "vlan100"
               set gateway 172.16.206.2
               set gateway6 2000:172:16:206::2
           next
       end
       config health-check
           edit "1"
               set addr-mode ipv6
               set server "2000::2:2:2:2"
   ```

```
            set members 0
            config sla
                edit 1
                next
            end
        next
    end
    config service
        edit 1
            set name "1"
            set addr-mode ipv6
            set mode sla
            set internet-service enable
            set internet-service-app-ctrl 16091
            set internet-service-app-ctrl-group "network-Ping"
            set internet-service-app-ctrl-category 15
            config sla
                edit "1"
                    set id 1
                next
            end
            set priority-members 4 1 2 3
        next
    end
end
```

2. View the default route for FGT_A:

```
config router static
    edit 5
        set distance 1
        set sdwan-zone "virtual-wan-link"
    next
end
```

3. View the firewall policy for FGT_A:

   The `utm-status` option is enabled to learn application 3T (3 tuple) information, and the default application profile of `g-default` is selected.

```
config firewall policy
    edit 1
        set uuid f09bddc4-def3-51ed-8517-0d8b6bc18f35
        set srcintf "any"
        set dstintf "any"
        set action accept
        set srcaddr6 "all"
        set dstaddr6 "all"
        set schedule "always"
        set service "ALL"
        set utm-status enable
        set ssl-ssh-profile "certificate-inspection"
        set application-list "g-default"
    next
end
```

To verify the configuration:

1. On FGT_A, check the routing table:

   The routing table has ECMP applied to default gateways for each SD-WAN member.

   ```
   # 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 IPSec-1 tunnel 172.16.209.2, [1/0]
   ```

2. Check the SD-WAN service:

   Based on the service rule, member 4 named vlan100 is preferred. Traffic must also match the highlighted internet services.

   ```
   # diagnose system sdwan service

   Service(1): Address Mode(IPV6) flags=0x4200 use-shortcut-sla use-shortcut
    Tie break: cfg
     Gen(2), TOS(0x0/0x0), Protocol(0: 1->65535), Mode(sla), sla-compare-order
     Members(4):
       1: Seq_num(4 vlan100), alive, sla(0x1), gid(0), cfg_order(0), local cost(0), 
   selected
       2: Seq_num(1 dmz), alive, sla(0x1), gid(0), cfg_order(1), local cost(0), selected
       3: Seq_num(2 IPSec-1), alive, sla(0x1), gid(0), cfg_order(2), local cost(0), 
   selected
       4: Seq_num(3 agg1), alive, sla(0x1), gid(0), cfg_order(3), local cost(0), selected
     Internet Service(3): Telnet(4294837974,0,0,0,0 16091) IPv6.ICMP(4294837087,0,0,0,0 
   16321) Network.Service(0,15,0,0,0)
   ```

3. Initiate traffic for ping, Telnet, and SSH to FGT_B, then FGT_A will learn 3T information for these applications, and use the SD-WAN rule to route traffic for the applications to the preferred interface of vlan100.

   - Following is the sniffer traffic for ping application. The ping traffic flows out of DMZ before 3T information is recognized, then out from vlan100 after T3 traffic is recognized:

     ```
     # diagnose sniffer packet any 'host 2000::2:0:0:4' 4
     interfaces=[any]
     filters=[host 2000::2:0:0:4]
     16.952138 port5 in 2000:172:16:205::100 -> 2000::2:0:0:4: icmp6: echo request seq 1 
     [flowlabel 0x5080d]
     16.954571 dmz out 2000:172:16:205::100 -> 2000::2:0:0:4: icmp6: echo request seq 1 
     [flowlabel 0x5080d]
     16.954920 dmz in 2000::2:0:0:4 -> 2000:172:16:205::100: icmp6: echo reply seq 1
     16.955086 port5 out 2000::2:0:0:4 -> 2000:172:16:205::100: icmp6: echo reply seq 1
     17.953277 port5 in 2000:172:16:205::100 -> 2000::2:0:0:4: icmp6: echo request seq 2 
     [flowlabel 0x5080d]
     17.953455 dmz out 2000:172:16:205::100 -> 2000::2:0:0:4: icmp6: echo request seq 2 
     [flowlabel 0x5080d]
     17.953622 dmz in 2000::2:0:0:4 -> 2000:172:16:205::100: icmp6: echo reply seq 2
     17.953722 port5 out 2000::2:0:0:4 -> 2000:172:16:205::100: icmp6: echo reply seq 2
     18.959823 port5 in 2000:172:16:205::100 -> 2000::2:0:0:4: icmp6: echo request seq 3 
     [flowlabel 0x5080d]
     18.960005 vlan100 out 2000:172:16:205::100 -> 2000::2:0:0:4: icmp6: echo request seq 
     3 [flowlabel 0x5080d]
     18.960015 agg1 out 2000:172:16:205::100 -> 2000::2:0:0:4: icmp6: echo request seq 3
     ```

```
[flowlabel 0x5080d]
18.960024 port4 out 2000:172:16:205::100 -> 2000::2:0:0:4: icmp6: echo request seq 3
[flowlabel 0x5080d]
18.960295 vlan100 in 2000::2:0:0:4 -> 2000:172:16:205::100: icmp6: echo reply seq 3
18.960449 port5 out 2000::2:0:0:4 -> 2000:172:16:205::100: icmp6: echo reply seq 3
19.983802 port5 in 2000:172:16:205::100 -> 2000::2:0:0:4: icmp6: echo request seq 4
[flowlabel 0x5080d]
```

- Following is the sniffer traffic for Telnet application group. The Telnet traffic flows out of agg1 before 3T information is recognized, then out from vlan100 after T3 traffic is recognized:

```
# diagnose sniffer packet any 'host 2000::2:0:0:4 and dst port 23' 4
interfaces=[any]
filters=[host 2000::2:0:0:4 and dst port 23]
4.096393 port5 in 2000:172:16:205::100.43128 -> 2000::2:0:0:4.23: syn 2723132265
[flowlabel 0xd4e65]
4.096739 agg1 out 2000:172:16:205::100.43128 -> 2000::2:0:0:4.23: syn 2723132265
[flowlabel 0xd4e65]
4.096752 port4 out 2000:172:16:205::100.43128 -> 2000::2:0:0:4.23: syn 2723132265
[flowlabel 0xd4e65]
.........
5.503679 port5 in 2000:172:16:205::100.43128 -> 2000::2:0:0:4.23: psh 2723132345 ack
544895389  [flowlabel 0xd4e65]
5.503894 vlan100 out 2000:172:16:205::100.43128 -> 2000::2:0:0:4.23: psh 2723132345
ack 544895389  [flowlabel 0xd4e65]
5.503907 agg1 out 2000:172:16:205::100.43128 -> 2000::2:0:0:4.23: psh 2723132345 ack
544895389  [flowlabel 0xd4e65]
5.503918 port4 out 2000:172:16:205::100.43128 -> 2000::2:0:0:4.23: psh 2723132345 ack
544895389  [flowlabel 0xd4e65]
5.504641 port5 in 2000:172:16:205::100.43128 -> 2000::2:0:0:4.23: ack 544895390
[flowlabel 0xd4e65]
5.504713 vlan100 out 2000:172:16:205::100.43128 -> 2000::2:0:0:4.23: ack 544895390
[flowlabel 0xd4e65]
5.504721 agg1 out 2000:172:16:205::100.43128 -> 2000::2:0:0:4.23: ack 544895390
[flowlabel 0xd4e65]
5.504728 port4 out 2000:172:16:205::100.43128 -> 2000::2:0:0:4.23: ack 544895390
[flowlabel 0xd4e65]
```

- Following is the sniffer traffic for SSH application category. The SSH traffic flows out of dmz before 3T information is recognized, then out from vlan100 after T3 traffic is recognized:

```
# diagnose sniffer packet any 'host 2000::2:0:0:4 and dst port 22' 4
interfaces=[any]
filters=[host 2000::2:0:0:4 and dst port 22]
5.910752 port5 in 2000:172:16:205::100.35146 -> 2000::2:0:0:4.22: syn 980547187
[flowlabel 0xf1403]
5.911002 dmz out 2000:172:16:205::100.35146 -> 2000::2:0:0:4.22: syn 980547187
[flowlabel 0xf1403]
5.914550 port5 in 2000:172:16:205::100.35146 -> 2000::2:0:0:4.22: ack 583860244
[flowlabel 0xf1403]
5.914651 dmz out 2000:172:16:205::100.35146 -> 2000::2:0:0:4.22: ack 583860244
[flowlabel 0xf1403]
.....
8.116507 port5 in 2000:172:16:205::100.35146 -> 2000::2:0:0:4.22: psh 980549261 ack
583862554  [class 0x10] [flowlabel 0xf1403]
8.116663 vlan100 out 2000:172:16:205::100.35146 -> 2000::2:0:0:4.22: psh 980549261
ack 583862554  [class 0x10] [flowlabel 0xf1403]
```

```
8.116674 agg1 out 2000:172:16:205::100.35146 -> 2000::2:0:0:4.22: psh 980549261 ack
583862554  [class 0x10] [flowlabel 0xf1403]
8.116685 port4 out 2000:172:16:205::100.35146 -> 2000::2:0:0:4.22: psh 980549261 ack
583862554  [class 0x10] [flowlabel 0xf1403]
8.118135 port5 in 2000:172:16:205::100.35146 -> 2000::2:0:0:4.22: ack 583862598
[class 0x10] [flowlabel 0xf1403]
8.118171 vlan100 out 2000:172:16:205::100.35146 -> 2000::2:0:0:4.22: ack 583862598
[class 0x10] [flowlabel 0xf1403]
8.118179 agg1 out 2000:172:16:205::100.35146 -> 2000::2:0:0:4.22: ack 583862598
[class 0x10] [flowlabel 0xf1403]
8.118189 port4 out 2000:172:16:205::100.35146 -> 2000::2:0:0:4.22: ack 583862598
[class 0x10] [flowlabel 0xf1403]
```

4. View the IPv6 application control internet service ID list:

```
# diagnose system sdwan internet-service-app-ctrl6-list

Telnet(16091 4294837974): 2000::2:0:0:4 6 23 Thu Apr 20 17:43:00 2023
IPv6.ICMP(16321 4294837087): 2000::2:0:0:4 58 0 Thu Apr 20 17:43:00 2023
```

5. View the IPv6 application control internet service ID list by category:

```
# diagnose system sdwan internet-service-app-ctrl6-category-list

SSH(16060 4294837772): 2000::2:0:0:4 6 22 Thu Apr 20 17:43:00 2023
```

Using a single IKE elector in ADVPN to match all SD-WAN control plane traffic

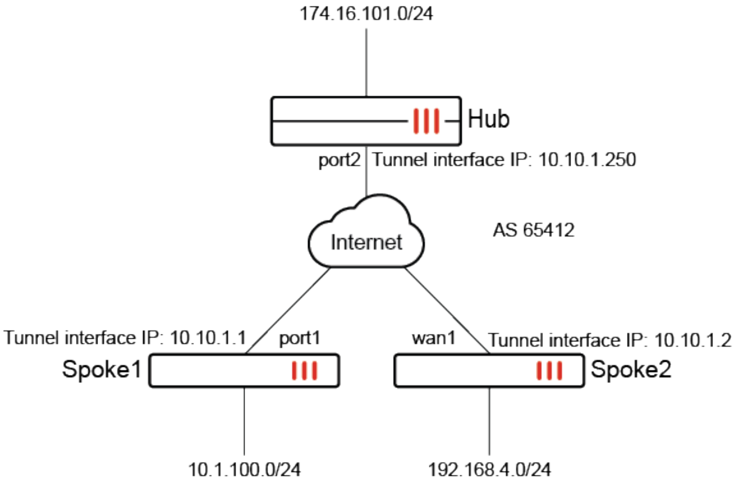
This information is also available in the FortiOS 7.4 Administration Guide:

- Using a single IKE elector in ADVPN to match all SD-WAN control plane traffic

In the SD-WAN with ADVPN use case, two spokes can communicate with each other on the control plane by an ADVPN shortcut. In order to separate the control traffic from data traffic, the IKE creates a dynamic selector for health check packets sent between the spokes. BGP traffic is also matched by this dynamic IKE selector. Therefore, when spokes establish BGP peering with other spokes, the BGP traffic does not count towards the data traffic and will not impact IPsec idle timeout and shortcut tunnel tear down.

Example

In this example, SD-WAN with ADVPN is configured. The IPsec ADVPN shortcut tunnel is required to tear down when it is idle. SD-WAN health checks are configured, and BGP neighbors established between the spokes is required.


174.16.101.0/24
Hub
port2
Tunnel interface IP: 10.10.1.250
Internet
AS 65412
Tunnel interface IP: 10.10.1.1
port1
wan1
Tunnel interface IP: 10.10.1.2
Spoke1
Spoke2
10.1.100.0/24
192.168.4.0/24

To configure the Hub FortiGate:

1. Configure the phase 1 interface:

```
config vpn ipsec phase1-interface
    edit "Hub"
        set type dynamic
        set interface "port2"
        set ike-version 2
        set peertype any
        set net-device disable
        set proposal aes128-sha256 aes256-sha256 aes128gcm-prfsha256 aes256gcm-prfsha384
chacha20poly1305-prfsha256
        set add-route disable
        set dpd on-idle
        set auto-discovery-sender enable
        set psksecret ************
        set dpd-retryinterval 60
    next
end
```

2. Configure the phase 2 interface:

```
config vpn ipsec phase2-interface
    edit "Hub"
        set phase1name "Hub"
        set proposal aes128-sha1 aes256-sha1 aes128-sha256 aes256-sha256 aes128gcm
aes256gcm chacha20poly1305
    next
end
```

3. Configure the VPN interface:

```
config system interface
    edit "Hub"
        set vdom "root"
        set ip 10.10.1.250 255.255.255.255
        set allowaccess ping
        set type tunnel
        set remote-ip 10.10.1.254 255.255.255.0
        set snmp-index 50
        set interface "port2"
    next
end
```

4. Configure the BGP settings:

```
config router bgp
    set as 65412
    config neighbor
        edit "10.10.1.1"
            set advertisement-interval 0
            set remote-as 65412
            set route-reflector-client enable
        next
        edit "10.10.1.2"
            set advertisement-interval 0
            set remote-as 65412
            set route-reflector-client enable
        next
    end
    config network
        edit 1
            set prefix 174.16.101.0 255.255.255.0
        next
    end
end
```

To configure the Spoke1 FortiGate:

1. Configure the phase 1 interface:

```
config vpn ipsec phase1-interface
    edit "Spoke1"
        set interface "port1"
        set ike-version 2
        set peertype any
        set net-device enable
        set proposal aes128-sha256 aes256-sha256 aes128gcm-prfsha256 aes256gcm-prfsha384 
chacha20poly1305-prfsha256
        set add-route disable
        set npu-offload disable
        set idle-timeout enable
        set idle-timeoutinterval 5
        set auto-discovery-receiver enable
        set remote-gw 172.16.200.4
        set psksecret ************
    next
end
```

2. Configure the phase 2 interface:

```
config vpn ipsec phase2-interface
    edit "Spoke1"
        set phase1name "Spoke1"
        set proposal aes128-sha1 aes256-sha1 aes128-sha256 aes256-sha256 aes128gcm 
aes256gcm chacha20poly1305
```

```
    next
end
```

3. Configure the VPN interface:

```
config system interface
    edit "Spoke1"
        set vdom "root"
        set ip 10.10.1.1 255.255.255.255
        set allowaccess ping
        set type tunnel
        set remote-ip 10.10.1.254 255.255.255.0
        set snmp-index 28
        set interface "port1"
    next
end
```

4. Configure the BGP settings:

```
config router bgp
    set as 65412
    config neighbor
        edit "10.10.1.250"
            set advertisement-interval 0
            set remote-as 65412
        next
        edit "10.10.1.2"
            set remote-as 65412
        next
    end
    config network
        edit 1
            set prefix 10.1.100.0 255.255.255.0
        next
    end
end
```

5. Configure the SD-WAN settings:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "Spoke1"
        next
    end
    config health-check
        edit "1"
            set server "174.16.101.44"
            set members 0
        next
    end
end
```

To configure the Spoke2 FortiGate:

1. Configure the phase 1 interface:

```
config vpn ipsec phase1-interface
    edit "Spoke2"
        set interface "wan1"
        set ike-version 2
        set peertype any
        set net-device enable
        set proposal aes128-sha256 aes256-sha256 aes128gcm-prfsha256 aes256gcm-prfsha384
chacha20poly1305-prfsha256
        set add-route disable
        set npu-offload disable
        set idle-timeout enable
        set idle-timeoutinterval 5
        set auto-discovery-receiver enable
        set remote-gw 172.16.200.4
        set psksecret ************
    next
end
```

2. Configure the phase 2 interface:

```
config vpn ipsec phase2-interface
    edit "Spoke2"
        set phase1name "Spoke2"
        set proposal aes128-sha1 aes256-sha1 aes128-sha256 aes256-sha256 aes128gcm
aes256gcm chacha20poly1305
    next
end
```

3. Configure the VPN interface:

```
config system interface
    edit "Spoke2"
        set vdom "root"
        set ip 10.10.1.2 255.255.255.255
        set allowaccess ping
        set type tunnel
        set remote-ip 10.10.1.254 255.255.255.0
        set snmp-index 15
        set interface "wan1"
    next
end
```

4. Configure the BGP settings:

```
config router bgp
    set as 65412
    config neighbor
        edit "10.10.1.250"
            set advertisement-interval 0
            set remote-as 65412
        next
        edit "10.10.1.1"
            set remote-as 65412
        next
    end
    config network
        edit 1
            set prefix 192.168.4.0 255.255.255.0
        next
    end
end
```

5. Configure the SD-WAN settings:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "Spoke2"
        next
    end
    config health-check
        edit "1"
            set server "174.16.101.44"
            set members 0
        next
    end
end
```

To verify the configuration:

1. Send traffic between the spokes to establish the ADVPN shortcut.
2. Verify the IPsec tunnel state on the Spoke1 FortiGate:

```
Spoke1 # diagnose vpn tunnel list
list all ipsec tunnel in vd 0
------------------------------------------------------
name=Spoke1_0 ver=2 serial=7 172.16.200.1:0->172.16.200.3:0 tun_id=10.10.1.2 tun_
id6=::10.0.0.3 dst_mtu=1500 dpd-link=on weight=1
bound_if=19 lgwy=static/1 tun=intf mode=dial_inst/3 encap=none/66224 options
[102b0]=create_dev rgwy-chg frag-rfc  role=primary accept_traffic=1 overlay_id=0

parent=Spoke1 index=0
proxyid_num=2 child_num=0 refcnt=6 ilast=0 olast=0 ad=r/2
stat: rxp=0 txp=1 rxb=0 txb=40
dpd: mode=on-demand on=1 idle=20000ms retry=3 count=0 seqno=1
natt: mode=none draft=0 interval=0 remote_port=0
fec: egress=0 ingress=0
proxyid=Spoke1 proto=0 sa=1 ref=5 serial=2 adr health-check
  src: 0:0.0.0.0-255.255.255.255:0
  dst: 0:10.10.1.2-10.10.1.2:0
  SA:  ref=3 options=92626 type=00 soft=0 mtu=1438 expire=43055/0B replaywin=2048
       seqno=214 esn=0 replaywin_lastseq=00000213 qat=0 rekey=0 hash_search_len=1
  life: type=01 bytes=0/0 timeout=43189/43200
  dec: spi=17a473be esp=aes key=16 40dfada9532cefe5563de71ac5908aa1
       ah=sha1 key=20 36e967d9b6fce8807132c3923d0edfae6cb6c115
  enc: spi=75cde30a esp=aes key=16 9bf08196d6830455a75bc676e04c816f
       ah=sha1 key=20 638db13dc4db0a6e5f523047805d18413eea4d4d
  dec:pkts/bytes=1060/42958, enc:pkts/bytes=1062/77075
  npu_flag=00 npu_rgwy=172.16.200.3 npu_lgwy=172.16.200.1 npu_selid=c dec_npuid=0 enc_
npuid=0
proxyid=Spoke1 proto=0 sa=1 ref=2 serial=1 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=12226 type=00 soft=0 mtu=1438 expire=43055/0B replaywin=2048
       seqno=2 esn=0 replaywin_lastseq=00000000 qat=0 rekey=0 hash_search_len=1
  life: type=01 bytes=0/0 timeout=43189/43200
  dec: spi=17a473bd esp=aes key=16 c78e5085857d0c5842e394fc44b38822
       ah=sha1 key=20 0bb885a85f77aa491a1209e4d36b7cddd7caf152
  enc: spi=75cde309 esp=aes key=16 6717935721e4a25428d6a7a633da75a9
       ah=sha1 key=20 eaf092280cf5b9f9db09ac95258786ffbfacead0
  dec:pkts/bytes=0/0, enc:pkts/bytes=2/144
  npu_flag=00 npu_rgwy=172.16.200.3 npu_lgwy=172.16.200.1 npu_selid=b dec_npuid=0 enc_
npuid=0
------------------------------------------------------
name=Spoke1 ver=2 serial=1 172.16.200.1:0->172.16.200.4:0 tun_id=172.16.200.4 tun_
id6=::172.16.200.4 dst_mtu=1500 dpd-link=on weight=1
bound_if=19 lgwy=static/1 tun=intf mode=auto/1 encap=none/560 options[0230]=create_dev
frag-rfc  role=primary accept_traffic=1 overlay_id=0

proxyid_num=1 child_num=1 refcnt=5 ilast=0 olast=0 ad=r/2
stat: rxp=542 txp=553 rxb=22117 txb=22748
dpd: mode=on-demand on=1 idle=20000ms retry=3 count=0 seqno=0
natt: mode=none draft=0 interval=0 remote_port=0
fec: egress=0 ingress=0
proxyid=Spoke1 proto=0 sa=1 ref=4 serial=1 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=12226 type=00 soft=0 mtu=1438 expire=42636/0B replaywin=2048
       seqno=22a esn=0 replaywin_lastseq=0000021f qat=0 rekey=0 hash_search_len=1
  life: type=01 bytes=0/0 timeout=42900/43200
  dec: spi=17a473bc esp=aes key=16 eff2dc03b48968bb55b9e3950ebde431
       ah=sha1 key=20 5db42a32aec15bc8a5fe392c256d1ae8ab3b4ef8
  enc: spi=bdc3bd80 esp=aes key=16 d0ec06b61ad572cc8813b599edde8c68
       ah=sha1 key=20 0306850f0184d957e9475da33d7971653a95c233
  dec:pkts/bytes=1084/44234, enc:pkts/bytes=1106/80932
  npu_flag=00 npu_rgwy=172.16.200.4 npu_lgwy=172.16.200.1 npu_selid=0 dec_npuid=0 enc_
npuid=0
```

The dynamic selector is created (highlighted) for SD-WAN control traffic, SD-WAN health checks, and BGP between spokes traffic.

3. Verify the BGP neighbors and check the routing table:

```
Spoke1 # get router info bgp summary

VRF 0 BGP router identifier 172.16.200.1, local AS number 65412
BGP table version is 8
1 BGP AS-PATH entries
0 BGP community entries

Neighbor    V         AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
10.10.1.2   4      65412      52      76        7    0    0 00:06:27        1
```

```
10.10.1.250 4      65412      70      69        1    0    0 00:58:44        2

Total number of neighbors 2
```

4. Stop sending traffic between the spokes, and wait for a few minutes (idle timeout).
5. Verify the IPsec tunnel state on the Spoke1 FortiGate:

```
Spoke1 # diagnose vpn tunnel list
list all ipsec tunnel in vd 0
--------------------------------------------------
name=Spoke1 ver=2 serial=1 172.16.200.1:0->172.16.200.4:0 tun_id=172.16.200.4 tun_
id6=::172.16.200.4 dst_mtu=1500 dpd-link=on weight=1
bound_if=19 lgwy=static/1 tun=intf mode=auto/1 encap=none/560 options[0230]=create_dev
frag-rfc  role=primary accept_traffic=1 overlay_id=0

proxyid_num=1 child_num=0 refcnt=4 ilast=0 olast=0 ad=r/2
stat: rxp=1467 txp=1469 rxb=60190 txb=60214
dpd: mode=on-demand on=1 idle=20000ms retry=3 count=0 seqno=0
natt: mode=none draft=0 interval=0 remote_port=0
fec: egress=0 ingress=0
proxyid=Spoke1 proto=0 sa=1 ref=3 serial=1 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=12226 type=00 soft=0 mtu=1438 expire=42199/0B replaywin=2048
       seqno=5be esn=0 replaywin_lastseq=000005bc qat=0 rekey=0 hash_search_len=1
  life: type=01 bytes=0/0 timeout=42903/43200
  dec: spi=76fdf7d1 esp=aes key=16 b26fd2dae76665f580d255b67f79df1e
       ah=sha1 key=20 14b0acc3c8c92a0af8ab43ff0437d2141b6d3f65
  enc: spi=bdc3bd85 esp=aes key=16 3eae3ad42aa32d7cdd972dfca286acd1
       ah=sha1 key=20 3655f67ee135f38e3f0790f1c7e3bd19c4a9285c
  dec:pkts/bytes=2934/120380, enc:pkts/bytes=2938/214606
  npu_flag=00 npu_rgwy=172.16.200.4 npu_lgwy=172.16.200.1 npu_selid=0 dec_npuid=0 enc_
npuid=0
```

   The shortcut tunnel between the spokes has been torn down. When data traffic is idle, the BGP traffic does not get sent on the data traffic selector, so the tunnel is not kept alive. This behavior is the expected, which consequently allows the shortcut tunnel to be torn down when idle.

6. Verify the IKE debugs messages to confirm the ADVPN shortcut was torn down:

```
Spoke1 # diagnose debug enable
Spoke1 # diagnose debug application ike -1
...
ike 0:Spoke1_0: connection idle time-out
ike 0:Spoke1_0: deleting
ike 0:Spoke1_0: flushing
ike 0:Spoke1_0: deleting IPsec SA with SPI 75cde338
ike 0:Spoke1_0:Spoke1: deleted IPsec SA with SPI 75cde338, SA count: 0
ike 0:Spoke1_0: sending SNMP tunnel DOWN trap for Spoke1
ike 0:Spoke1_0: tunnel down event 0.0.0.0
ike 0:Spoke1_0:Spoke1: delete
ike 0:Spoke1_0: deleting IPsec SA with SPI 75cde337
ike 0:Spoke1_0:Spoke1: deleted IPsec SA with SPI 75cde337, SA count: 0
ike 0:Spoke1_0: sending SNMP tunnel DOWN trap for Spoke1
ike 0:Spoke1_0: tunnel down event 0.0.0.0
ike 0:Spoke1_0:Spoke1: delete
ike 0:Spoke1_0: flushed
```

```
    ike 0:Spoke1_0:23:86: send informational
    ike 0:Spoke1_0:23: sent IKE msg (INFORMATIONAL): 172.16.200.1:500->172.16.200.3:500,
    len=80, vrf=0, id=0304e1284a432105/fa7d3fd75e7f481e:00000004
    ike 0:Spoke1_0: delete connected route 10.10.1.1 -> 10.10.1.2
    ike 0:Spoke1_0: delete dynamic
    ike 0:Spoke1_0: deleted
    ike 0:Spoke1: schedule auto-negotiate
    ike 0: comes 172.16.200.3:500->172.16.200.1:500,ifindex=19,vrf=0....
    ike 0: IKEv2 exchange=INFORMATIONAL_RESPONSE
    id=0304e1284a432105/fa7d3fd75e7f481e:00000004 len=80
```

VRF-aware SD-WAN IPv6 health checks

VRF and source can be configured in SD-WAN IPv6 health checks.

```
config system sdwan
   config health-check
      edit <name>
         set addr-mode ipv6
         set vrf <vrf id>
         set source6 <IPv6 address>
      next
   end
end
```

This example shows how to configure VRF and source for SD-WAN IPv6 health check on a standalone FortiGate.

To configure the VRF and source for SD-WAN IPv6 health check:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "R150"
            set gateway 10.100.1.1
            set gateway6 2000:10:100:1::1
        next
        edit 2
            set interface "R160"
            set gateway 10.100.1.5
            set gateway6 2000:10:100:1::5
        next
    end
    config health-check
        edit "ping6"
            set addr-mode ipv6
            set server "2000:10:100:2::22"
            set vrf 10
            set source6 2000:10:100:1::2
            set members 1 2
        next
    end
end
```

If an SD-WAN member can reach the server, but not on VRF 10, then it is dead:

```
# diagnose sys sdwan health-check
Health Check(ping6):
Seq(1 R150): state(alive), packet-loss(0.000%) latency(0.042), jitter(0.022), mos(4.404),
bandwidth-up(0), bandwidth-dw(0), bandwidth-bi(0) sla_map=0x0
Seq(2 R160): state(dead), packet-loss(100.000%) sla_map=0x0
```

Only the SD-WAN member with the proper VRF route can have the protocol 17 route, so the VRF is functioning correctly:

```
# diagnose ipv6 route list | grep protocol=17
vf=0 tbl=10 type=01(unicast) protocol=17(fortios) flag=00000000 prio=1024
src:2000:10:100:1::2/128-> dst:2000:10:100:2::22/128 gwy:2000:10:100:1::1 dev=48(R150)
pmtu=1500
```

Support maximize bandwidth (SLA) to load balance spoke-to-spoke traffic between multiple ADVPN shortcuts

This information is also available in the FortiOS 7.4 Administration Guide:

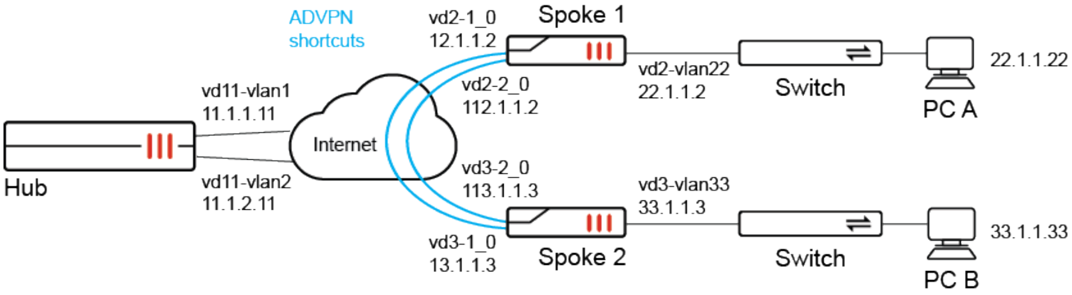
- Use maximize bandwidth to load balance traffic between ADVPN shortcuts

When ADVPN is configured on a FortiGate spoke along with an SD-WAN rule set to *Maximize Bandwidth SLA* (GUI) or load balance mode (CLI) as well as `tie-break` set to `fib-best-match`, then spoke-to-spoke traffic is load balanced between multiple ADVPN shortcuts when the shortcuts are within the configured SLA conditions.

Following is an example configuration with `set mode load-balance` and `set tie-break fib-best-match` enabled:

```
config system sdwan
    config service
        edit 3
            set mode load-balance
            set dst "all"
            config sla
                edit "ping"
                    set id 1
                next
            end
            set priority-members 1 2
            set tie-break fib-best-match
        next
    end
end
```

Example



In this example SD-WAN is configured between one hub and multiple spokes, and the SD-WAN configuration shows SD-WAN rule 3 with the following required settings to enable spoke-to-spoke traffic between multiple ADVPN shortcuts:

- set mode load-balance
- set tie-break fib-best-match

```
show system sdwan
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
        edit "zon2"
        next
    end
    config members
        edit 1
            set interface "vd2-1"
            set cost 10
        next
        edit 2
            set interface "vd2-2"
            set cost 20
        next
    end
    config health-check
        edit "ping"
            set server "11.11.11.11"
            set members 1 2
            config sla
                edit 1
                    set latency-threshold 200
                    set jitter-threshold 50
                next
                edit 2
                next
            end
        next
        edit "1"
        next
    end
    config service
        edit 1
```

```
            set dst "033"
            set priority-members 1
        next
        edit 2
            set dst "133"
            set priority-members 2
        next
        edit 3
            set mode load-balance
            set dst "all"
            config sla
                edit "ping"
                    set id 1
                next
            end
            set priority-members 1 2
            set tie-break fib-best-match
        next
    end
end
```

To trigger spoke-to-spoke communication, run an ICMP ping on PC A with IP address 22.1.1.22 behind spoke 1 that is destined for PC B with IP address 33.1.1.33 behind spoke 2. The spoke-to-spoke traffic will be used to demonstrate load balancing between shortcuts in the CLI output of this topic.

To verify the configuration:

1. Confirm the ADVPN shortcuts are within the SLA conditions:

```
# diagnose system sdwan health-check
Health Check(ping):
Seq(1 vd2-1): state(alive), packet-loss(0.000%) latency(0.029), jitter(0.002), mos
(4.404), bandwidth-up(1999), bandwidth-dw(0), bandwidth-bi(1999) sla_map=0x3
Seq(1 vd2-1_0): state(alive), packet-loss(0.000%) latency(0.026), jitter(0.001), mos
(4.404), bandwidth-up(2000), bandwidth-dw(0), bandwidth-bi(2000) sla_map=0x3
Seq(2 vd2-2): state(alive), packet-loss(0.000%) latency(0.055), jitter(0.064), mos
(4.404), bandwidth-up(0), bandwidth-dw(0), bandwidth-bi(0) sla_map=0x3
Seq(2 vd2-2_0): state(alive), packet-loss(0.000%) latency(0.060), jitter(0.058), mos
(4.404), bandwidth-up(0), bandwidth-dw(0), bandwidth-bi(0) sla_map=0x3
```

2. Confirm the settings for SD-WAN rule 3:

```
# diagnose system sdwan service 3

Service(3): Address Mode(IPV4) flags=0x4200 use-shortcut-sla use-shortcut
 Tie break: fib
  Gen(1), TOS(0x0/0x0), Protocol(0: 1->65535), Mode(load-balance  hash-mode=round-robin)
  Member sub interface(4):
    1: seq_num(1), interface(vd2-1):
       1: vd2-1_0(125)
    3: seq_num(2), interface(vd2-2):
       1: vd2-2_0(127)
  Members(4):
    1: Seq_num(1 vd2-1), alive, sla(0x1), gid(2), num of pass(1), selected
    2: Seq_num(1 vd2-1_0), alive, sla(0x1), gid(2), num of pass(1), selected
    3: Seq_num(2 vd2-2), alive, sla(0x1), gid(2), num of pass(1), selected
    4: Seq_num(2 vd2-2_0), alive, sla(0x1), gid(2), num of pass(1), selected
```

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

3. Confirm firewall policing routing list:

```
# diagnose firewall proute list 2131230723
list route policy info(vf=vd2):

id=2131230723(0x7f080003) vwl_service=3 vwl_mbr_seq=1 1 2 2 dscp_tag=0xfc 0xfc
flags=0x90 load-balance hash-mode=round-robin  fib-best-match tos=0x00 tos_mask=0x00
protocol=0 sport=0-65535 iif=0(any) dport=1-65535 path(4) oif=116(vd2-1) num_pass=1
oif=125(vd2-1_0) num_pass=1 oif=117(vd2-2) num_pass=1 oif=127(vd2-2_0) num_pass=1
destination(1): 0.0.0.0-255.255.255.255
source wildcard(1): 0.0.0.0/0.0.0.0
hit_count=117 last_used=2023-04-21 15:49:59
```

4. Confirm the routing table:

```
# get router info routing-table bgp
Routing table for VRF=0
B*      0.0.0.0/0 [200/0] via 10.10.100.254 (recursive via vd2-1 tunnel 11.1.1.11),
01:26:14, [1/0]
                  [200/0] via 10.10.200.254 (recursive via vd2-2 tunnel 11.1.2.11),
01:26:14, [1/0]
B       1.1.1.1/32 [200/0] via 11.1.1.1 [2] (recursive via 12.1.1.1, vd2-vlan12),
01:26:14, [1/0]
B       11.11.11.11/32 [200/0] via 10.10.100.254 (recursive via vd2-1 tunnel 11.1.1.11),
01:26:14, [1/0]
                       [200/0] via 10.10.200.254 (recursive via vd2-2 tunnel 11.1.2.11),
01:26:14, [1/0]
B       33.1.1.0/24 [200/0] via 10.10.100.3 [2] (recursive is directly connected, vd2-1_
0), 01:19:41, [1/0]
                    [200/0] via 10.10.200.3 [2] (recursive is directly connected, vd2-2_
0), 01:19:41, [1/0]
B       100.1.1.0/24 [200/0] via 10.10.100.254 (recursive via vd2-1 tunnel 11.1.1.11),
01:26:14, [1/0]
                     [200/0] via 10.10.200.254 (recursive via vd2-2 tunnel 11.1.2.11),
01:26:14, [1/0]
```

5. Check the packet sniffer output for the default setting.

   This step demonstrates routing for the default setting of `set tie-break zone`. The following packet sniffer output of ICMP pings demonstrates how spoke-to-spoke traffic (ping from 22.1.1.22 to 33.1.1.13) is load balanced between all parent tunnels and shortcuts, and is not limited to shortcuts within SLA.

```
# diagnose sniffer packet any "host 33.1.1.13" 4
interfaces=[any]
filters=[host 33.1.1.13]
14.665232 vd22-vlan22 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
14.665234 npu0_vlink1 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
14.665240 vd2-vlan22 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
14.665262 vd2-1_0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
14.665274 vd3-1_0 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
14.665284 vd3-vlan33 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
14.665285 npu0_vlink0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
14.665289 vd33-vlan33 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
14.665299 vd33-vlan33 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
14.665300 npu0_vlink1 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
```

```
14.665306 vd3-vlan33 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
14.665314 vd3-1_0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
14.665326 vd2-1_0 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
14.665331 vd2-vlan22 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
14.665332 npu0_vlink0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
14.665337 vd22-vlan22 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply

24.190955 vd22-vlan22 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.190957 npu0_vlink1 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.190963 vd2-vlan22 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.190982 vd2-2 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.190993 p2 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.191002 p2 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.191020 vd3-2 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.191031 vd3-vlan33 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.191032 npu0_vlink0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.191036 vd33-vlan33 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.191046 vd33-vlan33 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.191047 npu0_vlink1 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.191053 vd3-vlan33 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.191063 vd3-2 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.191074 p2 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.191079 p2 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.191090 vd2-2 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.191094 vd2-vlan22 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.191095 npu0_vlink0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.191100 vd22-vlan22 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply

51.064984 vd22-vlan22 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
51.064985 npu0_vlink1 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
51.064991 vd2-vlan22 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
51.065011 vd2-2_0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
51.065022 vd3-2_0 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
51.065031 vd3-vlan33 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
51.065032 npu0_vlink0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
51.065036 vd33-vlan33 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
51.065046 vd33-vlan33 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
51.065047 npu0_vlink1 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
51.065054 vd3-vlan33 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
51.065063 vd3-2_0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
51.065075 vd2-2_0 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
51.065082 vd2-vlan22 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
51.065082 npu0_vlink0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
51.065087 vd22-vlan22 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply

67.257123 vd22-vlan22 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
67.257125 npu0_vlink1 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
67.257131 vd2-vlan22 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
67.257150 vd2-1 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
67.257162 p1 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
67.257170 p1 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
67.257189 vd3-1 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
67.257199 vd3-vlan33 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
67.257200 npu0_vlink0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
67.257205 vd33-vlan33 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
67.257216 vd33-vlan33 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
```

```
67.257217 npu0_vlink1 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
67.257223 vd3-vlan33 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
67.257234 vd3-1 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
67.257245 p1 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
67.257250 p1 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
67.257261 vd2-1 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
67.257266 vd2-vlan22 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
67.257267 npu0_vlink0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
67.257272 vd22-vlan22 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply

^C
84 packets received by filter
0 packets dropped by kernel
```

6. Check the sniffer packet output after changing the setting to `set tie-break fib-best-match`.

   The following packet sniffer output of ICMP pings demonstrates how load balancing of spoke-to-spoke is limited and only occurs between shortcuts vd2-1_0 and vd2-2_0, which are within SLA.

```
# diagnose sniffer packet any "host 33.1.1.13" 4

interfaces=[any]
filters=[host 33.1.1.13]
2.592392 vd22-vlan22 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
2.592394 npu0_vlink1 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
2.592400 vd2-vlan22 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
2.592420 vd2-1_0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
2.592432 vd3-1_0 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
2.592441 vd3-vlan33 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
2.592442 npu0_vlink0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
2.592447 vd33-vlan33 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
2.592484 vd33-vlan33 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
2.592485 npu0_vlink1 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
2.592491 vd3-vlan33 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
2.592498 vd3-1_0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
2.592510 vd2-1_0 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
2.592515 vd2-vlan22 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
2.592516 npu0_vlink0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
2.592520 vd22-vlan22 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply

8.808792 vd22-vlan22 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.808793 npu0_vlink1 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.808799 vd2-vlan22 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.808816 vd2-2_0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.808827 vd3-2_0 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.808838 vd3-vlan33 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.808838 npu0_vlink0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.808842 vd33-vlan33 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.808852 vd33-vlan33 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
8.808853 npu0_vlink1 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
8.808858 vd3-vlan33 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
8.808866 vd3-2_0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
8.808877 vd2-2_0 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
8.808882 vd2-vlan22 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
8.808883 npu0_vlink0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
8.808887 vd22-vlan22 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
```

```
18.024377 vd22-vlan22 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
18.024379 npu0_vlink1 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
18.024385 vd2-vlan22 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
18.024400 vd2-1_0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
18.024411 vd3-1_0 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
18.024421 vd3-vlan33 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
18.024422 npu0_vlink0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
18.024427 vd33-vlan33 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
18.024436 vd33-vlan33 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
18.024437 npu0_vlink1 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
18.024443 vd3-vlan33 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
18.024449 vd3-1_0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
18.024459 vd2-1_0 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
18.024463 vd2-vlan22 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
18.024464 npu0_vlink0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
18.024468 vd22-vlan22 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply

24.216469 vd22-vlan22 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.216470 npu0_vlink1 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.216477 vd2-vlan22 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.216493 vd2-2_0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.216506 vd3-2_0 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.216518 vd3-vlan33 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.216519 npu0_vlink0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.216525 vd33-vlan33 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
24.216535 vd33-vlan33 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.216536 npu0_vlink1 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.216542 vd3-vlan33 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.216548 vd3-2_0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.216559 vd2-2_0 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.216563 vd2-vlan22 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.216564 npu0_vlink0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
24.216568 vd22-vlan22 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
^C
70 packets received by filter
0 packets dropped by kernel
```

7. Check SD-WAN heath.

   When an ADVPN shortcut is out of SLA, traffic does not run on it. Shortcut vd2-1_0 is out of SLA.

```
# diagnose system sdwan health-check
Health Check(ping):
Seq(1 vd2-1): state(alive), packet-loss(6.000%) latency(0.026), jitter(0.001), mos
(4.401), bandwidth-up(1999), bandwidth-dw(0), bandwidth-bi(1999) sla_map=0x0
Seq(1 vd2-1_0): state(alive), packet-loss(18.182%) latency(0.033), jitter(0.003), mos
(4.395), bandwidth-up(2000), bandwidth-dw(0), bandwidth-bi(2000) sla_map=0x0
Seq(2 vd2-2): state(alive), packet-loss(0.000%) latency(0.024), jitter(0.001), mos
(4.404), bandwidth-up(0), bandwidth-dw(0), bandwidth-bi(0) sla_map=0x3
Seq(2 vd2-2_0): state(alive), packet-loss(0.000%) latency(0.033), jitter(0.005), mos
(4.404), bandwidth-up(0), bandwidth-dw(0), bandwidth-bi(0) sla_map=0x3
```

8. Check the sniffer packet:

   No traffic runs on Shortcut vd2-1_0 because it is out of SLA.

```
# diagnose sniffer packet any "host 33.1.1.13" 4
interfaces=[any]
filters=[host 33.1.1.13]
```

```
8.723075 vd22-vlan22 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.723077 npu0_vlink1 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.723084 vd2-vlan22 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.723103 vd2-2_0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.723115 vd3-2_0 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.723148 vd3-vlan33 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.723149 npu0_vlink0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.723154 vd33-vlan33 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
8.723166 vd33-vlan33 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
8.723166 npu0_vlink1 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
8.723171 vd3-vlan33 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
8.723179 vd3-2_0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
8.723190 vd2-2_0 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
8.723195 vd2-vlan22 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
8.723195 npu0_vlink0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
8.723199 vd22-vlan22 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply

17.202681 vd22-vlan22 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
17.202683 npu0_vlink1 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
17.202688 vd2-vlan22 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
17.202704 vd2-2_0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
17.202716 vd3-2_0 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
17.202727 vd3-vlan33 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
17.202728 npu0_vlink0 out 22.1.1.22 -> 33.1.1.13: icmp: echo request
17.202733 vd33-vlan33 in 22.1.1.22 -> 33.1.1.13: icmp: echo request
17.202742 vd33-vlan33 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
17.202743 npu0_vlink1 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
17.202749 vd3-vlan33 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
17.202755 vd3-2_0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
17.202767 vd2-2_0 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
17.202771 vd2-vlan22 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
17.202772 npu0_vlink0 out 33.1.1.13 -> 22.1.1.22: icmp: echo reply
17.202777 vd22-vlan22 in 33.1.1.13 -> 22.1.1.22: icmp: echo reply
```

Allow multicast traffic to be steered by SD-WAN

This information is also available in the FortiOS 7.4 Administration Guide:

- Use SD-WAN rules to steer multicast traffic

SD-WAN rules can now steer multicast traffic. When an SD-WAN member is out of SLA, multicast traffic can fail over to another SD-WAN member, and switch back when SLA recovers.

The new `pim-use-sdwan` option enables or disables the use of SD-WAN for PIM (Protocol Independent Multicast) when checking RP (Rendezvous Point) neighbors and sending packets.

```
config router multicast
    config pim-sm-global
        set pim-use-sdwan {enable | disable}
    end
end
```

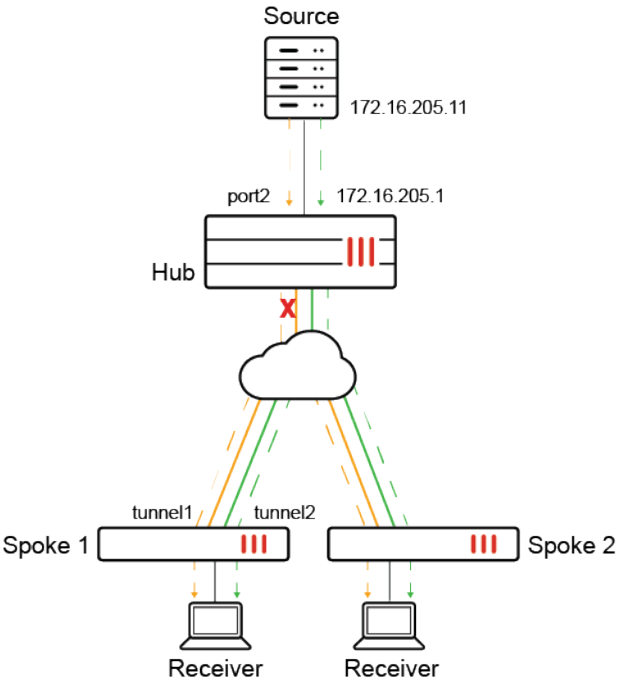
When SD-WAN steers multicast traffic, ADVPN is not supported. Use the `set shortcut` option to disable shortcuts for the service:

```
config system sdwan
    config service
        edit <id>
            set shortcut {enable | disable}
        next
    end
end
```

Example 1

In this hub and spoke example, the PIM source is behind the hub FortiGate, and the RP is set to internal port (port2) of the hub firewall. Each spoke connects to the two WAN interfaces on the hub by using an overlay tunnel. The overlay tunnels are members of SD-WAN.

Receivers behind the spoke FortiGates request a stream from the source to receive traffic on tunnel1 by default. When the overlay tunnel goes out of SLA, the multicast traffic fails over to tunnel2 and continues to flow.



Following is an overview of how to configure the topology:

1. Configure the hub FortiGate in front of the PIM source. The RP is configured on internal port (port2) of the hub FortiGate.
2. Configure the spoke FortiGates.
3. Verify traffic failover.

To configure the hub:

1. On the hub, enable multicast routing, configure the multicast RP, and enable PIM sparse mode on each interface:

```
config router multicast
    set multicast-routing enable
    config pim-sm-global
        config rp-address
            edit 1
                set ip-address 172.16.205.1
            next
        end
    end
    config interface
        edit "tport1"
            set pim-mode sparse-mode
        next
        edit "tagg1"
            set pim-mode sparse-mode
        next
        edit "port2"
            set pim-mode sparse-mode
        next
    end
end
```

To configure each spoke:

1. Enable SD-WAN with the following settings:
   - Configure the overlay tunnels as member of the SD-WAN zone.
   - Configure a performance SLA health-check using ping.
   - Configure a service rule for the PIM protocol with the following settings:
     - Use the lowest cost (SLA) strategy.
     - Monitor with the ping health-check.
   - Disable ADVPN shortcut.

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "tunnel1"
        next
        edit 2
            set interface "tunnel2"
        next
    end
    config health-check
        edit "ping"
            set server "172.16.205.1"
            set update-static-route disable
            set members 0
            config sla
                edit 1
                next
            end
        next
    end
    config service
        edit 1
            set mode sla
            set protocol 103
            set dst "all"
            config sla
                edit "ping"
                    set id 1
                next
            end
            set priority-members 1 2
            set use-shortcut-sla disable
            set shortcut disable
        next
        edit 2
            set mode sla
            set dst "all"
            config sla
                edit "ping"
                    set id 1
                next
            end
            set priority-members 1 2
        next
    end
end
```

2. Enable multicast routing and configure the multicast RP. Enable PIM sparse-mode on each interface:

```
config router multicast
    set multicast-routing enable
    config pim-sm-global
        set spt-threshold disable
        set pim-use-sdwan enable
        config rp-address
            edit 1
                set ip-address 172.16.205.1
            next
        end
    end
    config interface
        edit "tunnel1"
            set pim-mode sparse-mode
        next
        edit "tunnel2"
            set pim-mode sparse-mode
        next
        edit "port4"
            set pim-mode sparse-mode
        next
        end
    end
```

To verify traffic failover:

With this configuration, multicast traffic starts on tunnel1. When tunnel1 becomes out of SLA, traffic switches to tunnel2. When tunnel1 is in SLA again, the traffic switches back to tunnel1.

The following health-check capture on the spokes shows tunnel1 in SLA with packet-loss (1.000%):

```
# diagnose sys sdwan health-check
Health Check(ping):
Seq(1 tunnel1): state(alive), packet-loss(0.000%) latency(0.056), jitter(0.002), mos(4.404),
bandwidth-up(999999), bandwidth-dw(1000000), bandwidth-bi(1999999) sla_map=0x1
Seq(2 tunnel2): state(alive), packet-loss(0.000%) latency(0.100), jitter(0.002), mos(4.404),
bandwidth-up(0), bandwidth-dw(0), bandwidth-bi(0) sla_map=0x1

# diagnose sys sdwan health-check
Health Check(ping):
Seq(1 tunnel1): state(alive), packet-loss(1.000%) latency(0.056), jitter(0.002), mos(4.404),
bandwidth-up(999999), bandwidth-dw(1000000), bandwidth-bi(1999999) sla_map=0x1
Seq(2 tunnel2): state(alive), packet-loss(0.000%) latency(0.100), jitter(0.002), mos(4.404),
bandwidth-up(0), bandwidth-dw(0), bandwidth-bi(0) sla_map=0x1
```

The following example shows tunnel1 out of SLA with packet-loss (3.000%):

```
# diagnose sys sdwan health-check
Health Check(ping):
Seq(1 tunnel1): state(alive), packet-loss(3.000%) latency(0.057), jitter(0.003), mos(4.403),
bandwidth-up(999999), bandwidth-dw(1000000), bandwidth-bi(1999999) sla_map=0x0
Seq(2 tunnel2): state(alive), packet-loss(0.000%) latency(0.101), jitter(0.002), mos(4.404),
bandwidth-up(0), bandwidth-dw(0), bandwidth-bi(0) sla_map=0x1
```

The following example shows tunnel1 back in SLA again:

```
# diagnose sys sdwan health-check
Health Check(ping):
Seq(1 tunnel1): state(alive), packet-loss(1.000%) latency(0.061), jitter(0.004), mos(4.404),
bandwidth-up(999999), bandwidth-dw(1000000), bandwidth-bi(1999999) sla_map=0x0
Seq(2 tunnel2): state(alive), packet-loss(0.000%) latency(0.102), jitter(0.002), mos(4.404),
bandwidth-up(0), bandwidth-dw(0), bandwidth-bi(0) sla_map=0x1

# diagnose sys sdwan health-check
Health Check(ping):
Seq(1 tunnel1): state(alive), packet-loss(0.000%) latency(0.061), jitter(0.004), mos(4.404),
bandwidth-up(999999), bandwidth-dw(1000000), bandwidth-bi(1999999) sla_map=0x0
Seq(2 tunnel2): state(alive), packet-loss(0.000%) latency(0.102), jitter(0.002), mos(4.404),
bandwidth-up(0), bandwidth-dw(0), bandwidth-bi(0) sla_map=0x1
```

The following example how traffic switches to tunnel2 while tunnel1 health-check is out of SLA. Source (172.16.205.11) sends traffic to the multicast group. Later the traffic switches back to tunnel1 once SLA returns to normal:

```
195.060797 tunnel1 in 172.16.205.11 -> 225.1.1.1: icmp: echo request
195.060805 port4 out 172.16.205.11 -> 225.1.1.1: icmp: echo request
196.060744 tunnel1 in 172.16.205.11 -> 225.1.1.1: icmp: echo request
196.060752 port4 out 172.16.205.11 -> 225.1.1.1: icmp: echo request
```

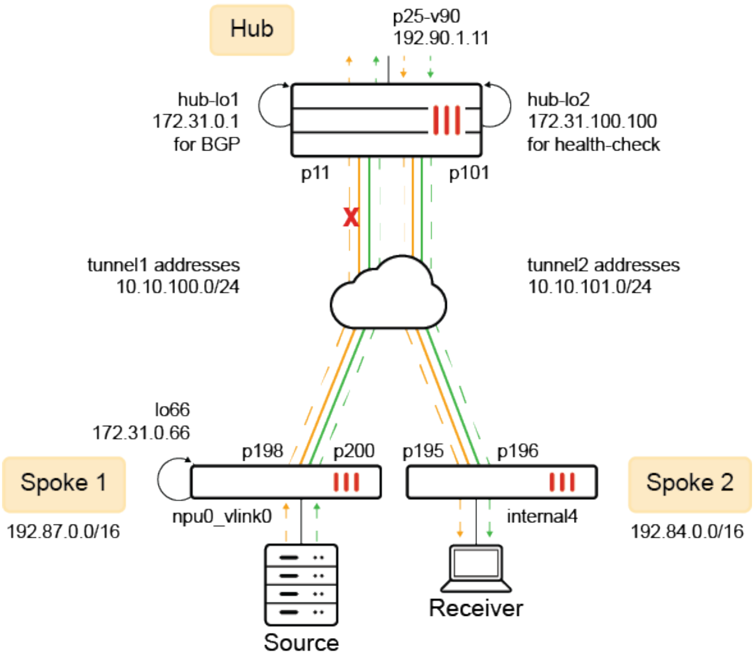
```
197.060728 tunnel1 in 172.16.205.11 -> 225.1.1.1: icmp: echo request
197.060740 port4 out 172.16.205.11 -> 225.1.1.1: icmp: echo request
198.060720 tunnel2 in 172.16.205.11 -> 225.1.1.1: icmp: echo request
198.060736 port4 out 172.16.205.11 -> 225.1.1.1: icmp: echo request
199.060647 tunnel2 in 172.16.205.11 -> 225.1.1.1: icmp: echo request
199.060655 port4 out 172.16.205.11 -> 225.1.1.1: icmp: echo request
200.060598 tunnel2 in 172.16.205.11 -> 225.1.1.1: icmp: echo request
200.060604 port4 out 172.16.205.11 -> 225.1.1.1: icmp: echo request
... ...
... ...
264.060974 port4 out 172.16.205.11 -> 225.1.1.1: icmp: echo request
265.060950 tunnel2 in 172.16.205.11 -> 225.1.1.1: icmp: echo request
265.060958 port4 out 172.16.205.11 -> 225.1.1.1: icmp: echo request
266.060867 tunnel2 in 172.16.205.11 -> 225.1.1.1: icmp: echo request
266.060877 port4 out 172.16.205.11 -> 225.1.1.1: icmp: echo request
267.060828 tunnel2 in 172.16.205.11 -> 225.1.1.1: icmp: echo request
267.060835 port4 out 172.16.205.11 -> 225.1.1.1: icmp: echo request
268.060836 tunnel1 in 172.16.205.11 -> 225.1.1.1: icmp: echo request
268.060854 port4 out 172.16.205.11 -> 225.1.1.1: icmp: echo request
269.060757 tunnel1 in 172.16.205.11 -> 225.1.1.1: icmp: echo request
269.060767 port4 out 172.16.205.11 -> 225.1.1.1: icmp: echo request
270.060645 tunnel1 in 172.16.205.11 -> 225.1.1.1: icmp: echo request
270.060653 port4 out 172.16.205.11 -> 225.1.1.1: icmp: echo request
```

Example 2

In this hub and spoke example, the PIM source is behind spoke 1, and the RP is configured on the hub FortiGate. BGP is used for routing. The hub uses embedded SLA in ICMP probes to determine the health of each tunnel, allowing it to prioritize healthy IKE routes.

The receiver is on another spoke. Upon requesting a stream, source passes the traffic to the RP on the hub FortiGate, and routes the traffic to the receiver over tunnel1. If a tunnel falls out of SLA, the multicast traffic fails over to the other tunnel.

In this configuration, SD-WAN steers multicast traffic by using embedded SLA information in ICMP probes. See also Embedded SD-WAN SLA information in ICMP probes. With this feature, the hub FortiGate can use the SLA information of the spoke's health-check to control BGP and IKE routes over tunnels.



Following is an overview of how to configure the topology:

1. Configure the hub FortiGate. The RP is configured on the hub FortiGate.
2. Configure the spoke FortiGate in front of the traffic receiver.
3. Configure the spoke FortiGate in front of the PIM source.

To configure the hub:

1. Configure loopbacks hub-lo1 172.31.0.1 for BGP and hub-lo100 172.31.100.100 for health-check:

```
config system interface
    edit "hub-lo1"
        set vdom "hub"
        set ip 172.31.0.1 255.255.255.255
        set allowaccess ping
        set type loopback
        set snmp-index 82
    next
    edit "hub-lo100"
        set vdom "hub"
        set ip 172.31.100.100 255.255.255.255
        set allowaccess ping
        set type loopback
        set snmp-index 81
    next
end
```

2. Enable multicast routing with the following settings:
   - Configure internal interface p25-v90 as RP.
   - Enable interfaces for PIM sparse-mode.

```
config router multicast
    set multicast-routing enable
    config pim-sm-global
        config rp-address
            edit 1
                set ip-address 192.90.1.11
            next
        end
    end
    config interface
        edit "p11"
            set pim-mode sparse-mode
        next
        edit "p101"
            set pim-mode sparse-mode
        next
        edit "p25-v90"
            set pim-mode sparse-mode
        next
    end
end
```

3. Enable SD-WAN with the following settings:
   - Add interfaces p11 and p101 as members.
   - Configure embedded SLA health-checks to detect ICMP probes from each overlay tunnel. Prioritize based on the health of each tunnel.

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "p11"
        next
        edit 2
            set interface "p101"
        next
    end
    config health-check
        edit "1"
            set detect-mode remote
            set probe-timeout 60000
            set recoverytime 1
            set sla-id-redistribute 1
            set members 1
            config sla
                edit 1
                    set link-cost-factor latency
                    set latency-threshold 100
                    set priority-in-sla 10
                    set priority-out-sla 20
                next
            end
        next
        edit "2"
```

```
                set detect-mode remote
                set probe-timeout 60000
                set recoverytime 1
                set sla-id-redistribute 1
                set members 2
                config sla
                    edit 1
                        set link-cost-factor latency
                        set latency-threshold 100
                        set priority-in-sla 15
                        set priority-out-sla 25
                    next
                end
            next
        end
    end
```

4. Configure BGP to peer with neighbors. Neighbor group is configured for tunnel interface IP addresses:

```
config router bgp
    set as 65505
    set router-id 172.31.0.1
    set ibgp-multipath enable
    set additional-path enable
    set recursive-inherit-priority enable
    config neighbor-group
        edit "gr1"
            set remote-as 65505
            set update-source "hub-lo1"
            set additional-path both
            set route-reflector-client enable
        next
    end
    config neighbor-range
        edit 1
            set prefix 10.10.0.0 255.255.0.0
            set neighbor-group "gr1"
        next
        edit 66
            set prefix 172.31.0.66 255.255.255.255
            set neighbor-group "gr1"
        next
    end
    config network
        ....
        edit 90
            set prefix 192.90.0.0 255.255.0.0
        next
    end
end
```

To configure the spoke (in front of the receiver):

1. Enable multicast routing to use SD-WAN. Configure the RP address. Enable interfaces for PIM sparse-mode.

```
config router multicast
    set multicast-routing enable
    config pim-sm-global
        set spt-threshold disable
        set pim-use-sdwan enable
        config rp-address
            edit 1
                set ip-address 192.90.1.11
            next
        end
    end
    config interface
        edit "p195"
            set pim-mode sparse-mode
        next
        edit "p196"
            set pim-mode sparse-mode
        next
        edit "internal4"
            set pim-mode sparse-mode
            set static-group "225-1-1-122"
        next
    end
end
```

2. Configure SD-WAN with the following settings:
    - Add overlay tunnel interfaces as members.
    - Configure a performance SLA health-check to send ping probes to the hub.
    - Configure a service rule for the PIM protocol. Use the lowest cost (SLA) strategy, and monitor with the ping health-check.
    - Disable ADVPN shortcuts.

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 6
            set interface "p196"
        next
        edit 5
            set interface "p195"
        next
    end
```

```
        config health-check
            edit "ping"
                set server "172.31.100.100"
                set update-static-route disable
                set members 0
                config sla
                    edit 1
                        set link-cost-factor latency
                        set latency-threshold 100
                    next
                end
            next
        end
        config service
            edit 1
                set mode sla
                set protocol 103
                set dst "all"
                config sla
                    edit "ping"
                        set id 1
                    next
                end
                set priority-members 5 6
                set use-shortcut-sla disable
                set shortcut disable
            next
            edit 2
                set mode sla
                set dst "all"
                config sla
                    edit "ping"
                        set id 1
                    next
                end
                set priority-members 5 6
            next
        end
    end
```

3. Configure BGP and set neighbors to the overlay gateway IP address on the hub:

```
config router bgp
    set as 65505
    set router-id 122.1.1.122
    set ibgp-multipath enable
    set additional-path enable
    config neighbor
        edit "10.10.100.254"
            set soft-reconfiguration enable
            set remote-as 65505
            set connect-timer 10
            set additional-path both
        next
        edit "10.10.101.254"
            set soft-reconfiguration enable
            set remote-as 65505
            set connect-timer 10
            set additional-path both
        next
    end
    config network
        edit 3
            set prefix 192.84.0.0 255.255.0.0
        next
    end
end
```

4. Configure the default gateway to use the SD-WAN zone. Other routes are for the underlay to route traffic to the hub's WAN interfaces:

```
config router static
    edit 10
        set distance 1
        set sdwan-zone "virtual-wan-link"
    next
    ....
    next
end
```

To configure the spoke (in front of the source):

1. Enable multicast routing to use SD-WAN. Configure the RP address. Enable interfaces for PIM sparse-mode:

```
config router multicast
    set multicast-routing enable
    config pim-sm-global
        set pim-use-sdwan enable
        config rp-address
            edit 1
                set ip-address 192.90.1.11
            next
        end
    end
    config interface
        edit "p198"
            set pim-mode sparse-mode
        next
        edit "p200"
            set pim-mode sparse-mode
        next
        edit "npu0_vlink0"
            set pim-mode sparse-mode
        next
    end
end
```

2. Configure loopback interface lo66 for BGP and sourcing SD-WAN traffic:

```
config system interface
    edit "lo66"
        set vdom "root"
        set ip 172.31.0.66 255.255.255.255
        set allowaccess ping
        set type loopback
        set snmp-index 21
    next
end
```

3. Configure SD-WAN:
   - Add overlay tunnel interfaces as members.
   - Configure a performance SLA health-check to send ping probes to the hub.
   - Configure a service rule for the PIM protocol. Use the lowest cost (SLA) strategy, and monitor with the ping health-check.
   - Disable the use of an ADVPN shortcut.

   In the following example, 11.11.11.11 is the underlay address for one of the WAN links on the hub, and 172.31.100.100 is the loopback address on the server.

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
        edit "overlay"
        next
    end
    config members
        edit 1
            set interface "p198"
            set zone "overlay"
            set source 172.31.0.66
        next
        edit 2
            set interface "p200"
            set zone "overlay"
            set source 172.31.0.66
        next
    end
    config health-check
        edit "ping"
            set server "11.11.11.11"
            set members 0
            config sla
                edit 1
                    set link-cost-factor latency
                    set latency-threshold 100
                next
            end
        next
        edit "HUB"
            set server "172.31.100.100"
            set embed-measured-health enable
            set members 0
            config sla
                edit 1
                    set link-cost-factor latency
                    set latency-threshold 100
                next
            end
        next
    end
     config service
        edit 1
            set mode sla
            set protocol 103
            set dst "all"
            config sla
                edit "ping"
                    set id 1
                next
            end
            set priority-members 1 2
            set use-shortcut-sla disable
            set shortcut disable
        next
        edit 2
            set mode sla
            set dst "all"
            config sla
                edit "ping"
                    set id 1
                next
            end
            set priority-members 1 2
        next
    end
end
```

4. Configure BGP:

```
config router bgp
    set as 65505
    set router-id 123.1.1.123
    set ibgp-multipath enable
    set additional-path enable
    config neighbor
        edit "172.31.0.1"
            set next-hop-self enable
            set soft-reconfiguration enable
            set remote-as 65505
            set update-source "lo66"
        next
    end
    config network
        edit 3
            set prefix 192.87.0.0 255.255.0.0
        next
    end
end
```

5. Configure the default gateway to use the SD-WAN zone. Other routes are for the underlay to route to the hub's WAN interfaces:

```
config router static
    edit 10
        set distance 1
        set sdwan-zone "virtual-wan-link" "overlay"
    next
    ...
    next
end
```

Support HTTPS performance SLA health checks - 7.4.1

HTTPS is supported for SD-WAN performance SLA health checks. All default HTTP-based health checks have been updated to use HTTPS instead. This includes:

- Default_AWS
- Default_FortiGuard
- Default_Google Search
- Default_Office_365

After upgrading, the default profiles using HTTP are changed to use HTTPS. Non-default performance SLA health check profiles are not affected after upgrading.

Example 1: applying a default HTTPS health check:

In this example, the Default_AWS health check is applied to an SD-WAN member in the default virtual-wan-link zone.

To apply the Default_AWS health check in an SD-WAN configuration:

1. Configure SD-WAN:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "port1"
            set gateway 172.16.200.254
            set gateway6 2000:172:16:200::254
        next
    end
    config health-check
        edit "Default_AWS"
            set server "aws.amazon.com"
            set protocol https
            set interval 1000
            set probe-timeout 1000
            set recoverytime 10
            set update-static-route disable
            set members 1
            config sla
                edit 1
                    set latency-threshold 250
                    set jitter-threshold 50
                    set packetloss-threshold 5
                next
            end
        next
    end
end
```

2. Verify the health check status:

```
# diagnose sys sdwan health-check status Default_AWS
Health Check(Default_AWS):
Seq(1 port1): state(alive), packet-loss(0.000%) latency(107.732), jitter(10.425), mos
(4.332), bandwidth-up(999920), bandwidth-dw(997555), bandwidth-bi(1997475) sla_map=0x1
```

Example 2: configuring an IPv6 health check with HTTPS

To configure an IPv6 health check with HTTPS:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "port1"
            set gateway 172.16.200.254
            set gateway6 2000:172:16:200::254
        next
    end
    config health-check
        edit "ipv6"
            set addr-mode ipv6
            set server "ipv6.google.com"
            set protocol https
            set members 1
            config sla
                edit 1
                    set latency-threshold 250
                    set jitter-threshold 50
```

```
                        set packetloss-threshold 5
                    next
                end
            next
        end
    end
```

Using load balancing in a manual SD-WAN rule without configuring an SLA target - 7.4.1

The maximize bandwidth (`load-balance`) strategy used prior to FortiOS 7.4.1 is now known as the load balancing strategy. This strategy can be configured under the manual mode and the lowest cost (SLA) strategies.

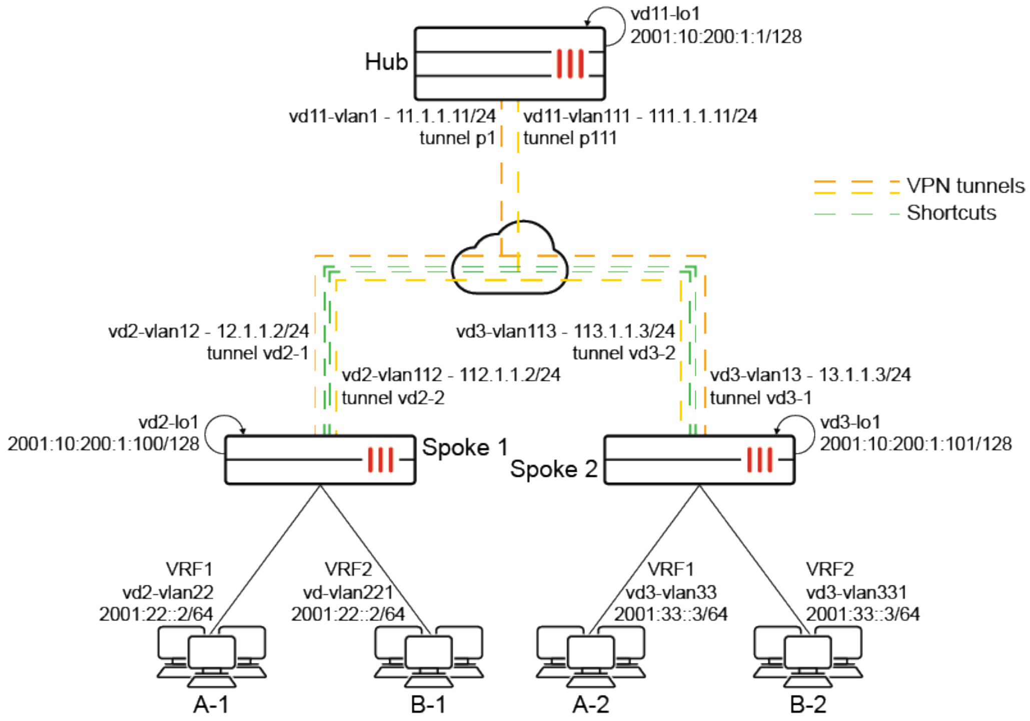
- When the load balancing strategy is configured under the manual mode strategy, SLA targets are not used.
- When the load balancing strategy is configured under the lowest cost (SLA) strategy, SLA targets are used.

IPv6 support for SD-WAN segmentation over a single overlay - 7.4.2

IPv6 is supported for SD-WAN segmentation over a single overlay. This allows seamless communication between IPv6 devices within virtual routing and forwarding (VRF) overlay networks, benefiting organizations transitioning to IPv6 or operating in a dual-stack environment.

Example

In this example, multiple companies (or departments of a company) share the ADVPN. Company A and company B each have two branches in two different locations. Company A's branches (A-1 and A-2) can talk to each other using the VPN shortcut, but not to company B's branches (B-1 and B-2). Likewise, company B's branches can talk to each other using the VPN shortcut, but not to company A's branches. Traffic can share the tunnels and shortcuts, but cannot be mixed up.



In this example, two spokes each have two tunnels to the hub.

- Each spoke has two VRFs behind it that can use the same IP address or subnets.
- The computers in VRF1 behind spoke 1 can talk to the computers in VRF1 behind spoke 2, but not to any of the computers in the VRF2s behind either spoke.
- The computers in VRF2 behind spoke 1 can talk to the computers in VRF2 behind spoke 2, but not to any of the computers in the VRF1s behind either spoke.
- Loopback addresses are used for communication between the spokes and the hub instead of tunnel IP address.

The `exchange-ip-addr6` option allows a loopback IPv6 address to be exchanged between the spokes and the hub in a network. This means that instead of using the tunnel IP address, which is typically used for communication, the loopback IPv6 address is used.

To configure the hub:

1. Configure the BGP settings:

```
config router bgp
    set as 65100
    set router-id 10.200.1.1
    set keepalive-timer 5
    set holdtime-timer 15
    set ibgp-multipath enable
    set network-import-check disable
    set additional-path6 enable
    set additional-path-vpnv6 enable
    set additional-path-select6 4
    config neighbor-group
        edit "EDGEv6"
```

```
            set advertisement-interval 1
            set activate disable
            set activate-vpnv4 disable
            set capability-graceful-restart enable
            set next-hop-self-rr6 enable
            set soft-reconfiguration6 enable
            set remote-as 65100
            set update-source "vd11-lo1"
            set additional-path6 both
            set adv-additional-path6 4
            set route-reflector-client6 enable
            set route-reflector-client-vpnv6 enable
        next
    end
    config neighbor-range6
        edit 2
            set prefix6 2001::10:200:1:0/112
            set neighbor-group "EDGEv6"
        next
    end
    config network6
        edit 1
            set prefix6 2001::10:200:1:0/112
        next
    end
    config vrf6
        edit "0"
            set role pe
        next
        edit "1"
            set role ce
            set rd "1:1"
            set export-rt "1:1"
            set import-rt "1:1"
        next
        edit "2"
            set role ce
            set rd "2:1"
            set export-rt "2:1"
            set import-rt "2:1"
        next
    end
end
```

2. Configure the IPsec phase 1 interface settings:

```
config vpn ipsec phase1-interface
    edit "p1"
        set type dynamic
        set interface "vd11-vlan1"
        set ike-version 2
        set peertype any
        set net-device disable
        set exchange-ip-addr6 2001::10:200:1:1
        set proposal aes128-sha256 aes256-sha256 aes128gcm-prfsha256 aes256gcm-prfsha384
chacha20poly1305-prfsha256
        set add-route disable
        set dpd on-idle
        set npu-offload disable
        set dhgrp 5
        set auto-discovery-sender enable
        set encapsulation vpn-id-ipip
        set psksecret **********
        set dpd-retryinterval 60
    next
    edit "p111"
        set type dynamic
        set interface "vd11-vlan111"
        set ike-version 2
        set peertype any
        set net-device disable
        set exchange-ip-addr6 2001::10:200:1:1
        set proposal aes128-sha256 aes256-sha256 aes128gcm-prfsha256 aes256gcm-prfsha384
chacha20poly1305-prfsha256
        set add-route disable
        set dpd on-idle
        set npu-offload disable
        set dhgrp 5
        set auto-discovery-sender enable
        set encapsulation vpn-id-ipip
        set psksecret **********
        set dpd-retryinterval 60
    next
end
```

3. Configure the IPsec phase 2 interface settings:

```
config vpn ipsec phase2-interface
    edit "p1-v6"
        set phase1name "p1"
        set proposal aes128-sha1
        set replay disable
        set src-addr-type subnet6
        set dst-addr-type subnet6
    next
    edit "p111-v6"
        set phase1name "p111"
        set proposal aes128-sha1
        set replay disable
        set src-addr-type subnet6
        set dst-addr-type subnet6
    next
end
```

To configure a spoke:

1. Configure the BGP settings:

```
config router bgp
    set as 65100
    set router-id 10.200.1.100
    set keepalive-timer 5
    set holdtime-timer 15
    set ibgp-multipath enable
```

```
        set additional-path6 enable
        set additional-path-vpnv6 enable
        set recursive-next-hop enable
        set tag-resolve-mode merge
        set graceful-restart enable
        set additional-path-select6 4
        config neighbor
            edit "2001::10:200:1:1"
                set advertisement-interval 1
                set activate disable
                set activate-vpnv4 disable
                set capability-dynamic enable
                set capability-graceful-restart6 enable
                set capability-graceful-restart-vpnv6 enable
                set soft-reconfiguration6 enable
                set remote-as 65100
                set route-map-in6 "tag"
                set route-map-in-vpnv6 "tag"
                set connect-timer 10
                set update-source "vd2-lo1"
                set additional-path6 both
                set additional-path-vpnv6 both
            next
        end
        config network6
            edit 1
                set prefix6 2001:22::/64
            next
            edit 2
                set prefix6 2001::10:200:1:100/128
            next
        end
        config vrf6
            edit "0"
                set role pe
            next
            edit "1"
                set role ce
                set rd "1:1"
                set export-rt "1:1"
                set import-rt "1:1"
            next
            edit "2"
                set role ce
                set rd "2:1"
                set export-rt "2:1"
                set import-rt "2:1"
            next
        end
    end
```

2. Configure the IPsec phase 1 interface settings:

```
config vpn ipsec phase1-interface
    edit "vd2-1"
        set interface "vd2-vlan12"
        set ike-version 2
        set peertype any
        set net-device enable
        set exchange-ip-addr6 2001::10:200:1:100
        set proposal aes128-sha256 aes256-sha256 aes128gcm-prfsha256 aes256gcm-prfsha384
chacha20poly1305-prfsha256
        set add-route disable
        set npu-offload disable
        set dhgrp 5
        set auto-discovery-receiver enable
        set encapsulation vpn-id-ipip
        set remote-gw 11.1.1.11
        set psksecret **********
    next
    edit "vd2-2"
        set interface "vd2-vlan112"
        set ike-version 2
        set peertype any
        set net-device enable
        set exchange-ip-addr6 2001::10:200:1:100
        set proposal aes128-sha256 aes256-sha256 aes128gcm-prfsha256 aes256gcm-prfsha384
chacha20poly1305-prfsha256
        set add-route disable
        set npu-offload disable
        set dhgrp 5
        set auto-discovery-receiver enable
        set encapsulation vpn-id-ipip
        set remote-gw 111.1.1.11
        set psksecret **********
    next
end
```

3. Configure the IPsec phase 2 interface settings:

```
config vpn ipsec phase2-interface
    edit "vd2-1-6"
        set phase1name "vd2-1"
        set proposal aes128-sha1
        set dhgrp 5
        set replay disable
        set auto-negotiate enable
        set src-addr-type subnet6
        set dst-addr-type subnet6
    next
    edit "vd2-2-6"
        set phase1name "vd2-2"
        set proposal aes128-sha1
        set dhgrp 5
        set replay disable
        set auto-negotiate enable
        set src-addr-type subnet6
        set dst-addr-type subnet6
    next
end
```

4. Configure the SD-WAN settings:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "vd2-1"
            set cost 10
        next
        edit 2
            set interface "vd2-2"
            set cost 20
        next
    end
    config health-check
        edit "ping6"
            set addr-mode ipv6
            set server "2001::10:200:1:1"
            set source6 2001::10:200:1:100
            set members 1 2
            config sla
                edit 1
                next
            end
        next
    end
    config service
        edit 61
            set addr-mode ipv6
            set priority-members 1
            set dst6 "6001-100"
        next
        edit 62
            set addr-mode ipv6
            set priority-members 2
            set dst6 "6100-200"
        next
    end
end
```

To check the spoke 1 routes:

```
# get router info6 routing-table bgp
Routing table for VRF=0
B       2001::10:200:1:0/112 [200/0] via 2001::10:200:1:1 tag 100 (recursive via vd2-1 
tunnel ::11.1.1.11), 1d15h41m
(recursive via vd2-2 tunnel ::111.1.1.11), 1d15h41m, [1024/0]
B       2001::10:200:1:101/128 [200/0] via 2001::10:200:1:1 tag 100 (recursive via vd2-1 
tunnel ::11.1.1.11), 1d15h41m
(recursive via vd2-2 tunnel ::111.1.1.11), 1d15h41m, [1024/0]

Routing table for VRF=1
B V     2001:33::/64 [200/0] via 2001::10:200:1:101 tag 100 (recursive via vd2-1 tunnel 
::11.1.1.11), 1d15h41m
```

```
(recursive via vd2-2 tunnel ::111.1.1.11), 1d15h41m, [1024/0]

Routing table for VRF=2
B V     2001:33::/64 [200/0] via 2001::10:200:1:101 tag 100 (recursive via vd2-1 tunnel
::11.1.1.11), 1d15h41m
(recursive via vd2-2 tunnel ::111.1.1.11), 1d15h41m, [1024/0]
```

To test the configuration on shortcut 1:

1. From VRF1 of spoke 1, ping VRF1 of spoke 2.
2. From VRF2 of spoke 1, ping VRF2 spoke 2. Both VRF1 and VRF2 source and destination IP addresses are the same, so you can see how the traffic is isolated.
3. Verify the session list:

```
# diagnose sys session6 list
session6 info: proto=58 proto_state=00 duration=3 expire=59 timeout=0 refresh_dir=both
flags=00000000 sockport=0 socktype=0 use=3
origin-shaper=
reply-shaper=
per_ip_shaper=
class_id=0 ha_id=0 policy_dir=0 tunnel=/ vlan_cos=0/0
state=may_dirty
statistic(bytes/packets/allow_err): org=416/4/0 reply=416/4/0 tuples=2
tx speed(Bps/kbps): 136/1 rx speed(Bps/kbps): 136/1
orgin->sink: org pre->post, reply pre->post dev=100->223/223->100
hook=pre dir=org act=noop 2001:22::55:398->2001:33::44:128(:::0)
hook=post dir=reply act=noop 2001:33::44:398->2001:22::55:129(:::0)
src_mac=02:4c:a5:fc:77:6f
misc=0 policy_id=1 pol_uuid_idx=1070 auth_info=0 chk_client_info=0 vd=3:2
serial=0001104d tos=ff/ff ips_view=0 app_list=0 app=0 url_cat=0
sdwan_mbr_seq=0 sdwan_service_id=61
rpdb_link_id=ff00003d ngfwid=n/a
npu_state=0x1040001 no_offload
no_ofld_reason:  disabled-by-policy non-npu-intf
total session6: 1
```

In the output, `vd=<vdom_ID>:<VRF_ID>` indicates that sessions are created in and stay in the corresponding VRFs.

BGP incorporates the advanced security measures of TCP Authentication Option (TCP-AO) - 7.4.2

Border Gateway Protocol (BGP) incorporates the advanced security measures of TCP Authentication Option (TCP-AO), which supports stronger algorithms, such as AES-128 CMAC and HMAC-SHA1. This integration bolsters the security of and enhances the reliability of BGP connections and contributes to the overall security of the internet.

CLI changes include:

- Added `cmac-aes128` option in the router key-chain:

```
config router key-chain
    edit <name>
```

```
        config key
            edit <id>
                set algorithm cmac-aes128
            next
        end
    next
end
```

- Added `auth-options` for BGP `neighbor` and `neighbor-group`:

```
config router bgp
    config neighbor|neighbor-group
        edit <string>
            set auth-options <string>
        end
    next
end
```

- Added debug command for `tcp-auth-options`:

```
diagnose sys tcp-auth-options
```

Example

In this example, the router BGP neighbor is configured to use the AES-128 CMAC algorithm.

To configure the router BGP to use the AES-128 CMAC algorithm:

1. Configure the router key-chain to use the AES-128 CMAC algorithm:

```
config router key-chain
    edit "11"
        config key
            edit "1"
                set accept-lifetime 01:01:01 01 01 2021 2147483646
                set send-lifetime 01:01:01 01 01 2021 2147483646
                set key-string **********
                set algorithm cmac-aes128
            next
        end
    next
end
```

2. Apply the key-chain to the BGP neighbor or neighbor group:
In this example, the key-chain is applied to the BGP neighbor with IP address 2.2.2.2.

```
config router bgp
    set as 65412
    config neighbor
        edit "2.2.2.2"
            set auth-options "11"
        next
    end
end
```

3. Verify that the router BGP is using the algorithm.
The command output shows that BGP neighbor 2.2.2.2 is using the AES-128 CMAC algorithm.

```
# diagnose sys tcp-auth-options

VFID=0 send-id=1 recv-id=1 flags=0x784 keylen=6
alg=2(aes128) addr=2.2.2.2
send-begin: Fri Jan  1 01:01:01 2021
send-end: Wed Jan 19 04:15:07 2089
recv-begin: Fri Jan  1 01:01:01 2021
recv-end: Wed Jan 19 04:15:07 2089
```

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