

Campus Fabric DHCP Relay—Juniper Validated Design Extension (JVDE)

Published
2025-05-23

Table of Contents

About this Document	1
Solution Benefits	1
Solution Architecture	2
Validation Framework	15
Test Objectives	18
Recommendations	19
APPENDIX: Example Microsoft DHCP Server Configuration Used for Testing	22
APPENDIX: Example Linux KEA DHCP Server Configuration Used for Testing	23
APPENDIX: Example DHCP Relay in EVPN Multihoming Fabric	27
APPENDIX: Example DHCP Relay in IP Clos Fabric	50
APPENDIX: Fabrics with Overlay Loopback Usage Before June 20, 2024, Deployed	61

Campus Fabric DHCP Relay—Juniper Validated Design Extension (JVDE)

Juniper Networks Validated Designs provide customers with a comprehensive, end-to-end blueprint for deploying Juniper solutions in their network. These designs are created by Juniper's expert engineers and tested to ensure they meet the customer's requirements. Using a validated design, customers can reduce the risk of costly mistakes, save time and money, and ensure that their network is optimized for maximum performance.

About this Document

This document covers how to successfully implement DHCP relay with all existing campus fabric types. Part of the integration of almost all campus fabric installations is the need to provide a scalable and redundant way to assign DHCP leases to wired and wireless clients. Most customers design and use their own DHCP servers that need to be integrated into the design as they also provide integration into other systems such as DNS or security functions such as Network Access Control (NAC). In this case, the campus fabric is instructed to forward the DHCP lease response through a DHCP relay function from the customer-provided DHCP server responding to the original lease request of the client. The reader of this JVD will find the different architectures and forwarding options explained and how they operate. The appendix of this JVD contains two example configurations from two major third-party DHCP server vendors and configuration examples for two fabric types tested.

Solution Benefits

In enterprise networks, it is mission critical that devices such as Juniper® Series of High-Performance Access Points and attached wired and wireless clients obtain DHCP leases from the network without difficulty. This critical functionality requires a DHCP server somewhere in the network to manage the lease handouts. In most cases, customers have existing DHCP servers from third-party vendors that need to be integrated when designing and installing a new campus fabric. In these cases, the campus fabric itself is not responsible for providing the DHCP server functionality. Instead, the fabric must forward the DHCP lease requests from clients attached to access switches from the fabric to the customer-provided DHCP server to manage the lease and respond back to the client through the fabric. The fabric relays DHCP traffic between DHCP client and server by providing DHCP relay functionality. We share more information about this process and how it works later in this chapter.

The recommended production-grade solution is the [IETF RFC 2131](#) and [RFC 3046](#) standards-based approach leveraging DHCP relay inside the fabric to forward to a customer-managed DHCP server. For such a production-grade design utilizing DHCP relay in the fabric, we suggest choosing a third-party DHCP server that can support the following:

- Provide at least two DHCP server instances for redundancy of this critical function. It is beneficial if the redundant servers share a common database for the lease handouts. Otherwise, you must split the ranges handed out for a particular VLAN between the servers as each then requires a unique non-overlapping pool range.
- For DHCP relay to work properly, the DHCP server is required to listen on an IP address-based socket interface as the traditional Layer 2 broadcast listening won't work here. More details explained below.
- OPTIONAL: Provide an interface and integration with your DNS server.
- OPTIONAL: Provide an interface and integration with your security functions such as a NAC when required to manage or monitor new clients in the network.

The Junos OS-based fabric switches themselves either do not provide this functionality or they do so in a limited fashion.

Solution Architecture

IN THIS SECTION

- [Proof of Concept Non-Production-Grade DHCP Server Integration Approach | 3](#)
- [Recommended Production-Grade DHCP Server Integration Approach | 4](#)
- [Virtual Gateway Fabric Versus Anycast Fabric | 6](#)
- [Which IP Address to Choose as Reported Gateway IP Address? | 7](#)
- [Where to Locate the DHCP Server | 8](#)
- [What Needs to be Considered When Using External DHCP Servers? | 10](#)
- [Optimizations in Junos OS to Help Your DHCP Relay Design | 13](#)

Before mentioning the suggested production-grade architecture, for completion's sake, we will share the approach to use if security is not a concern and faster results are preferred.

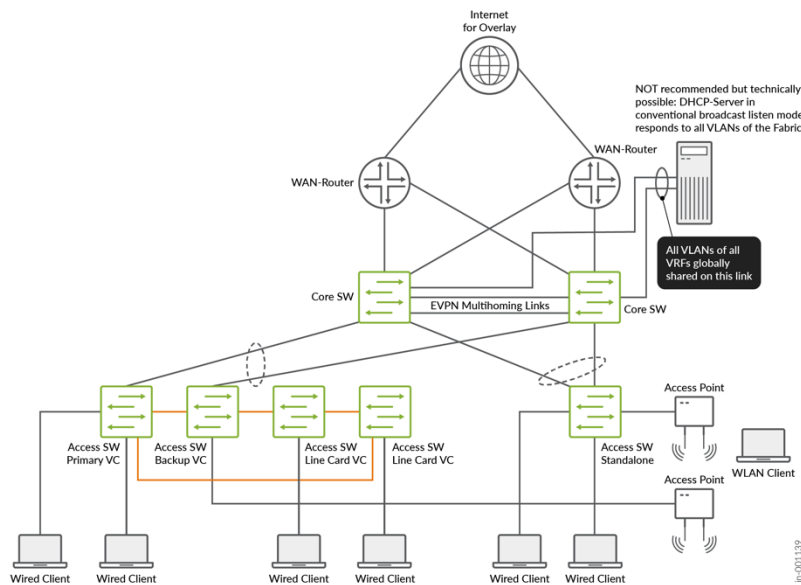
Proof of Concept Non-Production-Grade DHCP Server Integration Approach

In this approach, the DHCP server is attached locally to the fabric's service block function. For redundancy, an ESI-LAG is configured on the fabric side and a normal LAG on the server side. All possible 4,000+ VLANs are configured and exposed on this link towards the DHCP server. The fabric itself then needs no extra configuration of DHCP relay as the Layer 2 broadcast domains of all access VLANs are stretched to the DHCP server. By configuring matching sub-interfaces to listen on, the DHCP server can then assign leases by listening to the MAC broadcast packets the clients send. This works the old-fashioned way, and many DHCP servers (including Junos OS) can respond to this traffic.

NOTE: This is clearly not recommended in production-grade designs and rollouts. It may help getting the network up faster, but there are severe security risks that come with such a design!

The reason for not recommending this in production-grade designs is that you bypass a basic security function of the fabric design. Normally, all fabric VRFs are isolated from each other. This forces all traffic between VLANs in different VRFs to transit the WAN router where security functions such as firewalls can be implemented. By placing all VLANs on the link to the DHCP server in the service block, we bypass the WAN router and thus the security functionality. If an attacker finds a security hole such as an open administrator login, or something is misconfigured, the attacker can jump between all VLANs and maybe also bypass any WAN router-based screening between VRFs. **Please be aware of this trade-off.**

Figure 1: EVPN Multihoming with 2 Collapsed Cores



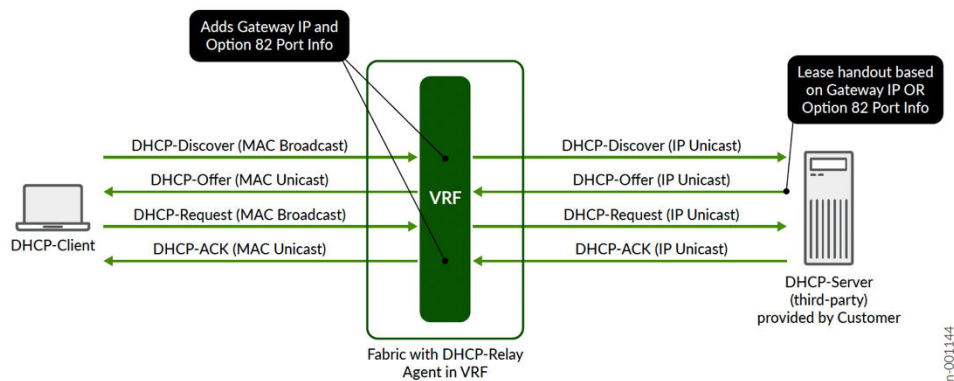
Recommended Production-Grade DHCP Server Integration Approach

The recommended production-grade solution is to follow an [IETF RFC 2131](#) and [RFC 3046](#) standards-based approach leveraging DHCP relay inside the fabric to forward to a customer-managed DHCP server.

In a campus fabric, the DHCP relay is usually located where the Layer 3 gateway VRF sends traffic from the attached access switches towards the north-bound functions. This is because this is the demarcation between the Layer 2 broadcast domain and the remaining Layer 3 forwarding in the network. A DHCP client requesting a lease always addresses the broadcast MAC address in this Layer 2 domain, which is FF:FF:FF:FF:FF:FF. Once such a message reaches the VRF, it is forwarded by the VRF's DHCP relay function (RFC 2131 calls this a BOOTP relay agent) as an IP UDP packet towards the DHCP server attached to the fabric or outside of it. Along with this, forwarding additional information is always embedded in the UDP packet towards the DHCP server:

- The Gateway IP (RFC 2131 calls this the “giaddr”) is the source IP address of the DHCP packet. The DHCP servers send all response packets back to this embedded IP address. Hence, it is important that this IP address is reachable by the responding packets. This reported source IP may differ depending on the fabric type and the WAN router integration method. The two most popular methods which are illustrated in this JVD are:
 - For small fabrics using virtual gateway addressing, the static IP address for each overlay VLAN that is locally configured on each fabric VRF is used as the gateway IP.

- For larger fabrics where anycast addressing is in use, an additional range of overlay loopback IPs is used by the VRFs distributed across the fabric. Each VRF across all nodes where VRFs are configured in this fabric gets a unique IP address used as the gateway IP. Hence, you have an additional overlay network, but one that is shared across VRFs. It needs to be noted that in this case there are two flavours of loopback IP address usage:
 - There are always the existing underlay loopback IP addresses bound to the lo0.0 interface on each EVPN fabric switch. These are used for VXLAN VTEP and EVPN BGP signalling. They are typically invisible for the overlay transport.
 - The new loopback IP addresses used for DHCP relay are visible in the overlay VLANs. When defining this overlay loopback pool, take care that this range is not used elsewhere by the overlay VLANs. Also, the IP addresses in this overlay loopback pool must only be bound to nodes of the EVPN fabric where the VRFs are located.
- It's expected that the DHCP relay function embeds additional information such as that added by DHCP option 82 according to RFC 3046. This helps the DHCP server to identify the source of the client request and to be able to better manage the lease handout decision. This is needed because the DHCP server will no longer be able to receive DHCP lease requests from local VLANs since it is configured to listen on an IP address socket-interface receiving UDP packets. The possible attributes that can be embedded in option 82 by the DHCP server vendor can vary and is always subject to integration with the fabric.



As already mentioned, the fabric configures the DHCP relay function where the VRFs are located in your fabric. Do not attempt to manually change this on the switches. Inside the fabric configuration fields, you will find DHCP relay configuration options in the pane where you configure the networks and VRFs. The rest will be deployed automatically by the Juniper Mist™ cloud on the necessary nodes as indicated in the figure below:

Fabric Type	DHCP Relay will be Configured On	Number of Supported Nodes
EVPN Multihoming	Collapsed Core Switches	Maximum 4
CRB	Core Switches	Maximum 4
ERB	Distribution Switches	many
IP Clos Routed at Edge	Access Switches	many

Virtual Gateway Fabric Versus Anycast Fabric

Depending on the fabric type, the overlay VLANs (where the client traffic is located), may need additional IP addresses for internal purposes, which is the case for virtual gateway fabrics. The Juniper Mist campus fabric configures the following fabric types:

Fabric Type	Virtual Gateway Fabric	Anycast Fabric
EVPN Multihoming	Yes	---
CRB	Yes	---
ERB	---	Yes
IP Clos Fabric	---	Yes

In a virtual gateway fabric, you typically have a very limited amount of VRFs. Those are located on the core or collapsed core switches. The maximum amount of core or collapsed core switches supported in a Juniper Mist campus fabric is four. This means a certain VRF can be duplicated to each redundant core or collapsed core switch a maximum of four times in the fabric. Anycast fabrics, as opposed to virtual gateway fabrics, are appropriate for more scaled designs. Hence, the location of the VRFs is either on the distribution switches (ERB) or at the access switch (IP Clos fabric). The nature of virtual gateway fabrics is that the system requires an additional static IP address that is unique per VRF for every VLAN located in the fabric. Hence, in addition to the shared gateway IP address for each VLAN, up to four additional unique IP addresses on that subnet are required.

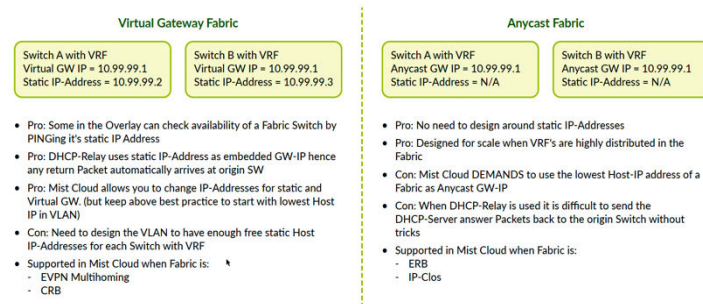
Why such a design? There are benefits for certain traffic on the fabric such as DHCP relay. For DHCP relay, the system uses the static IP address instead of the gateway IP address when forwarding the

DHCP client requests. This behavior ensures that the DHCP response packet will be sent back to the correct VRF since the static IP address is unique to the VLAN/core switch.

Another way to think about a virtual gateway fabric is if you compare it with traditional Layer 2 gateway failover designs such as VRRP. There you always have a VIP which floats between the gateways (that are our VRFs) and each gateway needs an additional unique static IP for each VLAN. In a Juniper Mist campus fabric, the VRRP protocol is not needed as the EVPN control plane takes over for it.

The small sacrifice needed to carve out those additional static IP addresses in each subnet is eliminated in anycast fabrics. This is because of the more scaled distribution/access switches where VRFs are installed you would have to plan your future growth well when creating VLANs. System services such as DHCP relay work in anycast fabrics a bit differently and are internally more complex.

Figure 2: Virtual Gateway Versus Anycast Fabric Types



Which IP Address to Choose as Reported Gateway IP Address?

You must typically choose between one of the following approaches to determine which gateway IP address is embedded in the packets forwarded by the DHCP relay function in the fabric:

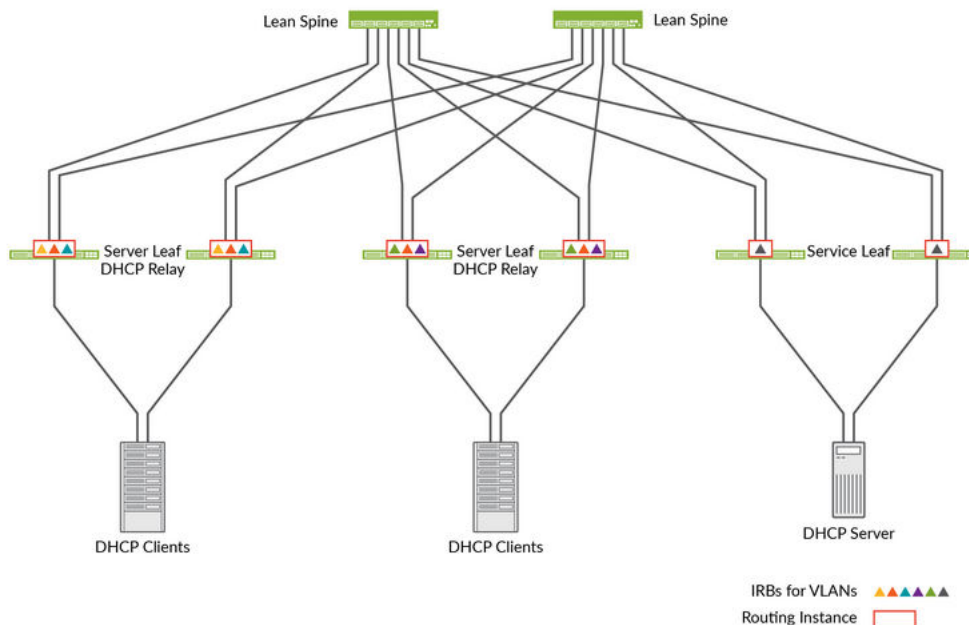
- For EVPN multihoming fabrics, the UI does not provide any choice so you will always be using virtual gateway IP addresses for the gateway IP. This enables the DHCP server to identify the VLAN where the request originates by only analyzing the gateway IP address embedded in the forwarded packets.
- For CRB fabrics, you can select between virtual gateway static IP address design by leaving the field "Loopback per-VRF subnet" empty or by using a design with overlay loopback IP addresses assigned to each VRF in the fabric when populating an IP prefix in the "Loopback per-VRF subnet" of the campus fabric dialogue.
- For larger fabrics such as ERB and IP Clos, we recommend entering an IP prefix in the "Loopback per-VRF subnet" as part of the campus fabric configuration. By doing so, the fabric will automatically

assign unique overlay loopback IPs out of this pool range to each VRF in the fabric. In this case, we also highly recommend leveraging a routing protocol such as OSPF or BGP for WAN router integration towards the fabric as the usage of these overlay loopback IPs can make it difficult to predict how each gateway IP can be reached from the WAN router.

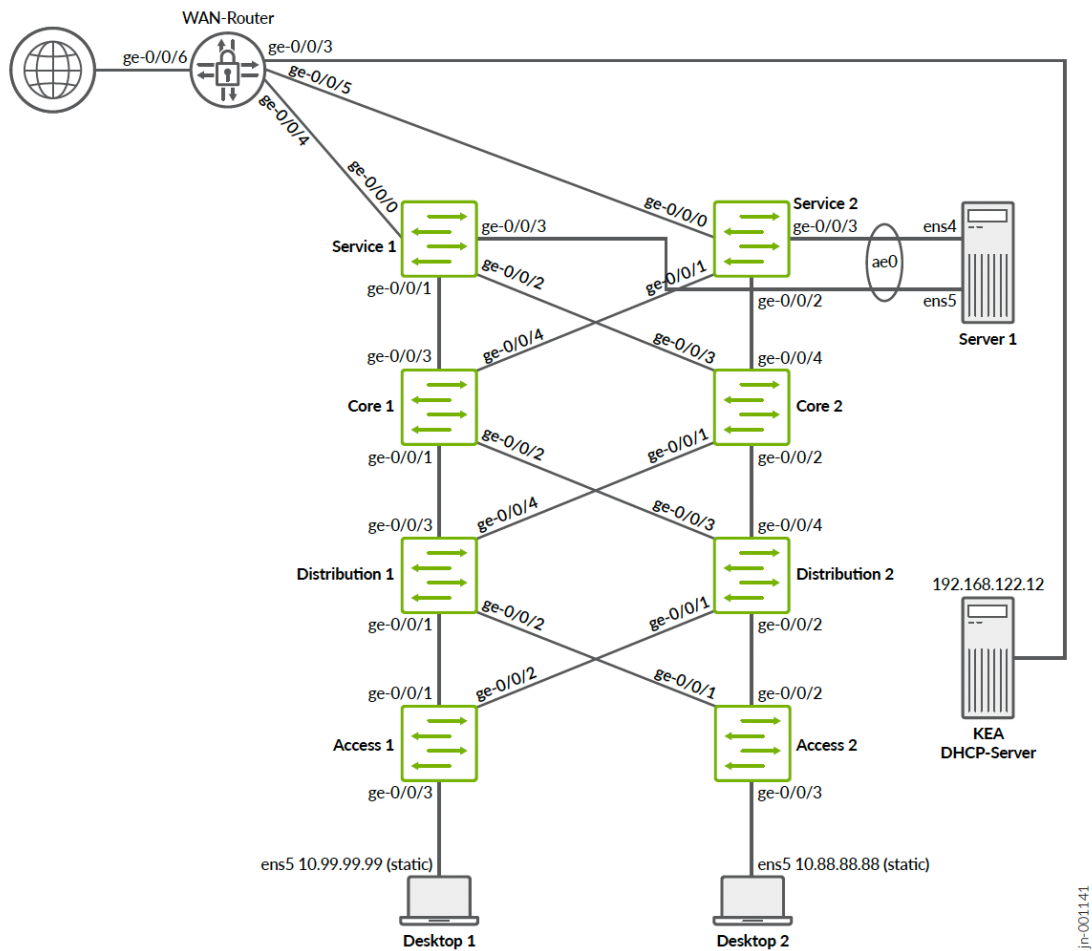
While it is technically possible to leave the "Loopback per-VRF subnet" field empty in these fabrics, it is not recommended. If left blank, the anycast gateway IP will be the reported gateway IP embedded in the forwarded packets. This doesn't cause issues on the way to the DHCP server. However, when the DHCP server's response returns, because the anycast IP is shared by multiple switches within the fabric, the response packet might be routed to a switch that did not originate the request and could be in a different PoD or building. If this happens, when the packet arrives at a switch that didn't initiate the request, the DHCP relay function will decapsulate the response and determine, based on the client's MAC address, that the packet must be forwarded to a remote switch. To do so, it will use a VXLAN tunnel to resend the packet east-west to the switch where the client is actually connected. This creates an inefficient design.

Where to Locate the DHCP Server

The approach we recommend is local integration as part of the fabric itself.

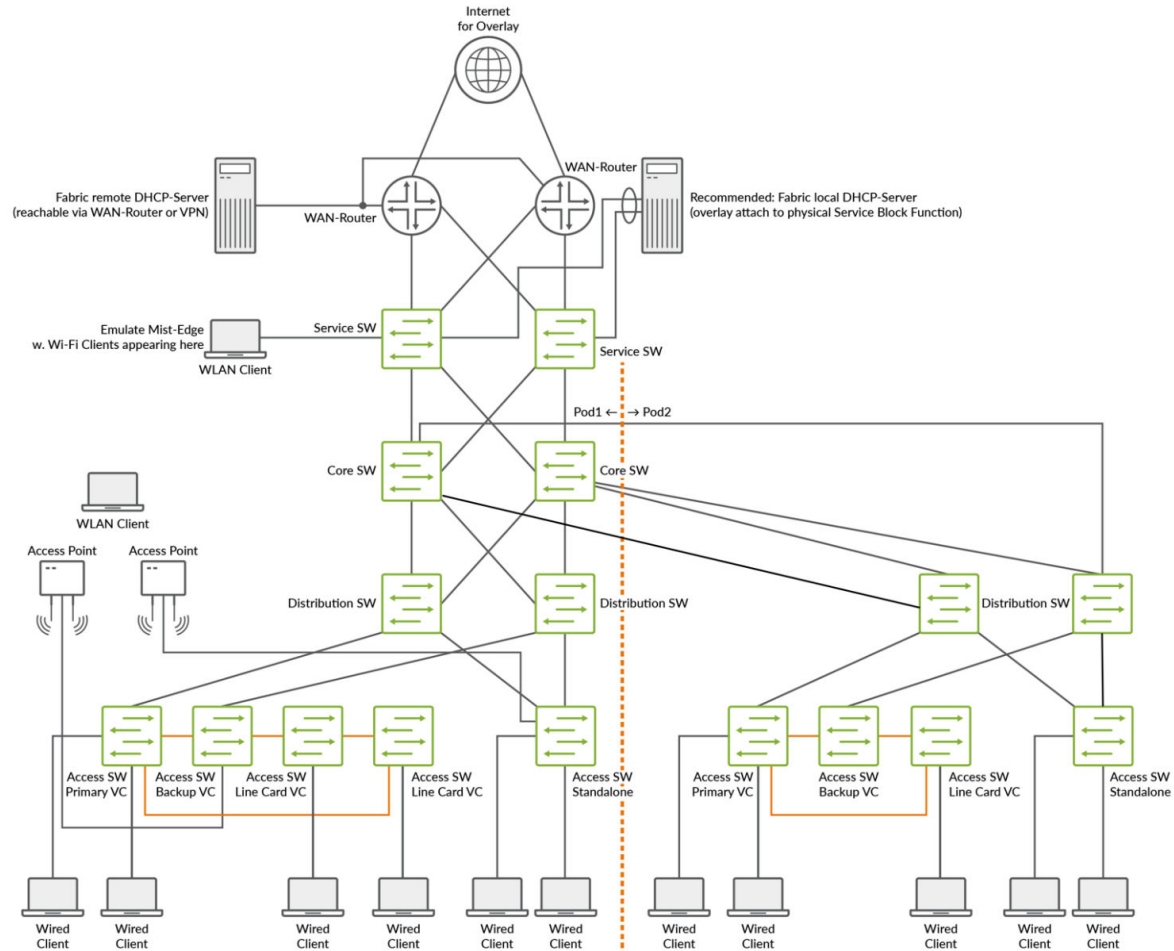


In the case of a campus fabric, the service leaf is called a service block function and, in the following example, a pair of physical switches north of where the DHCP server is attached to be an integral piece of the fabric.



The DHCP server can also manage leases for clients not in the same VRF as itself. However, as there is always VRF to VRF isolation inside the fabric, the packets will have to be exchanged through a WAN router.

When such a local integration is not possible, one can also try to integrate the DHCP server as an external element towards the fabric. This is what you see in the upper-left corner in the following figure:



What Needs to be Considered When Using External DHCP Servers?

When integrating external DHCP servers into the fabric, there are two critical points to keep in mind for this approach to succeed:

- **Keep the latency between the fabric and DHCP servers as low as possible.** Do not consider operating the DHCP server in a public cloud environment if you do not know the latency impact. Some DHCP clients can be very aggressive when requesting a lease, which can cause unanswered DHCP lease requests to stack up in a high latency environment, overloading the DHCP server. Since the DHCP client behavior is hardly influenceable at the fabric level, we recommend testing the design with a focus on the entire round-trip latency times before putting the fabric into production.

- **No form of network address translation (NAT) is supported between the fabric and the DHCP server.** The reason for this is explained in the following excerpt from [IETF RFC 2131](#) describing how a DHCP server must respond to client requests. Remember, “giaddr” is the embedded gateway IP address:

```
4.1 Constructing and sending DHCP messages
.
If the 'giaddr' field in a DHCP message from a client is non-zero,
the server sends any return messages to the 'DHCP server' port on the
BOOTP relay agent whose address appears in 'giaddr'.
.
```

You may not be aware what this RFC description means, so we've provided an example of traffic using a Microsoft DHCP server that does not respond to SNATed DHCP requests in accordance with the RFC.

Here is the original source IP and port from the access switch created for the relay packet:

```
tcpdump -nnvvi fabric3 'port 4789 and udp[8:2] = 0x0800 and udp[11:4] = 11099'
tcpdump: listening on fabric3, link-type EN10MB (Ethernet), capture size 262144 bytes
12:48:50.799126 IP (tos 0x0, ttl 63, id 0, offset 0, flags [DF], proto UDP (17), length 419)
    172.16.254.6.42357 > 172.16.254.1.4789: [no cksum] VXLAN, flags [I] (0x08), vni 11099
IP (tos 0x0, ttl 64, id 27564, offset 0, flags [none], proto UDP (17), length 369)
    10.99.99.1.67 > 192.168.10.11.67: [udp sum ok] BOOTP/DHCP, Request from 52:54:00:d8:d1:04,
length 341, xid 0x892af3d, Flags [none] (0x0000)
    Gateway-IP 10.99.99.1
    Client-Ethernet-Address 52:54:00:d8:d1:04
    Vendor-rfc1048 Extensions
        Magic Cookie 0x63825363
        DHCP-Message Option 53, length 1: Discover
        Hostname Option 12, length 8: "desktop1"
        Parameter-Request Option 55, length 13:
            Subnet-Mask, BR, Time-Zone, Default-Gateway
            Domain-Name, Domain-Name-Server, Option 119, Hostname
            Netbios-Name-Server, Netbios-Scope, MTU, Classless-Static-Route
            NTP
        Agent-Information Option 82, length 39:
            Circuit-ID SubOption 1, length 4: 1099
            Unknown SubOption 9, length 31:
                0x0000: 0000 0a4c 1a04 1849 5242 2d69 7262 2e31
                0x0010: 3039 393a 6d67 652d 302f 302f 332e 30
```

When the WAN router applies the SNAT to the forwarded discover message, it changes the source IP and port as follows:

```
tcpdump -nnvvi br0 'host 192.168.10.11'
12:40:46.545512 IP (tos 0x0, ttl 63, id 5475, offset 0, flags [none], proto UDP (17), length 369)
    192.168.10.169.28594 > 192.168.10.11.67: [udp sum ok] BOOTP/DHCP, Request from
52:54:00:d8:d1:04, length 341, xid 0x78caf527, secs 7, Flags [none] (0x0000)
    Gateway-IP 10.99.99.1
    Client-Ethernet-Address 52:54:00:d8:d1:04
    Vendor-rfc1048 Extensions
        Magic Cookie 0x63825363
        DHCP-Message Option 53, length 1: Discover
        Hostname Option 12, length 8: "desktop1"
        Parameter-Request Option 55, length 13:
            Subnet-Mask, BR, Time-Zone, Default-Gateway
            Domain-Name, Domain-Name-Server, Option 119, Hostname
            Netbios-Name-Server, Netbios-Scope, MTU, Classless-Static-Route
            NTP
        Agent-Information Option 82, length 39:
            Circuit-ID SubOption 1, length 4: 1099
            Unknown SubOption 9, length 31:
                0x0000: 0000 0a4c 1a04 1849 5242 2d69 7262 2e31
                0x0010: 3039 393a 6d67 652d 302f 302f 332e 30
```

The DHCP server's response, however, answers back to the original embedded gateway IP on port 67, which was the original source IP inside the fabric before the SNAT was applied:

```
12:40:46.545962 IP (tos 0x0, ttl 128, id 30987, offset 0, flags [none], proto UDP (17), length 359)
    192.168.10.11.67 > 10.99.99.1.67: [bad udp cksum 0x397c -> 0x81a7!] BOOTP/DHCP, Reply,
length 331, xid 0x78caf527, Flags [none] (0x0000)
    Your-IP 10.99.99.10
    Server-IP 192.168.10.11
    Gateway-IP 10.99.99.1
    Client-Ethernet-Address 52:54:00:d8:d1:04
    Vendor-rfc1048 Extensions
        Magic Cookie 0x63825363
        DHCP-Message Option 53, length 1: Offer
        Subnet-Mask Option 1, length 4: 255.255.255.0
        RN Option 58, length 4: 345600
        RB Option 59, length 4: 604800
```

```

Lease-Time Option 51, length 4: 691200
Server-ID Option 54, length 4: 192.168.10.11
Default-Gateway Option 3, length 4: 10.99.99.1
Domain-Name-Server Option 6, length 8: 8.8.8.8,9.9.9.9
Agent-Information Option 82, length 39:
  Circuit-ID SubOption 1, length 4: 1099
  Unknown SubOption 9, length 31:
    0x0000: 0000 0a4c 1a04 1849 5242 2d69 7262 2e31
    0x0010: 3039 393a 6d67 652d 302f 302f 332e 30

```

This packet will never arrive back at the SNAT firewall.

```

root@wanrouter> show security flow session destination-prefix 192.168.10.11
Session ID: 22116, Policy name: default-permit/5, Timeout: 50, Valid
  In: 10.99.99.1/67 --> 192.168.10.11/67;udp, Conn Tag: 0x0, If: ae0.1099, Pkts: 72, Bytes:
26568,
  Out: 192.168.10.11/67 --> 192.168.10.169/6673;udp, Conn Tag: 0x0, If: ge-0/0/0.0, Pkts: 0,
Bytes: 0,

```

Optimizations in Junos OS to Help Your DHCP Relay Design

Through the Junos OS configuration, a couple of statements help to optimize your design. We present the critical ones here in case you want to know why the fabric is automatically configuring those:

In the Junos OS configuration, a “forward-only” statement is needed to prevent the device from monitoring uncontrolled DHCP traffic.

```

set groups top routing-instances <vrf-name> forwarding-options dhcp-relay forward-only

```

In the Junos OS configuration, the “relay-option-82 circuit-id vlan-id-only” statement synchronizes the behavior of QFX and EX switches when forwarding option 82 DHCP traffic (by default, they use different attributes). Also, this attribute then no longer adds unwanted interface information such as “IRB-irb.1099:ae10.0” or “IRB-irb.1099:vtep.32769”. With this configuration added, only the VLAN ID is

reported, easing the string parsing in this field, which we leverage for the Linux KEA DHCP server to assign the right lease.

```
set groups top routing-instances <vrf-name> forwarding-options dhcp-relay group <vlan-name>
relay-option-82 circuit-id vlan-id-only
```

The Junos OS configuration statement “relay-option-82 server-id override” is described [here](#) and is needed in our environment. It helps the Microsoft DHCP server using sub-option 5 to determine the source of the packet and choose the right pool to assign.

```
set groups top routing-instances <vrf-name> forwarding-options dhcp-relay group <vlan-name>
relay-option-82 server-id override
```

The Junos OS configuration statement “route-suppression destination” is needed when using loopback IPs as gateway IPs. It is not used for virtual gateway address static IPs as gateway IPs.

```
set groups top routing-instances <vrf-name> forwarding-options dhcp-relay group <vlan-name>
route-suppression destination
```

In the Junos OS configuration, the fabric configures all IRB interfaces now with the “no-dhcp-flood” option. This helps to limit the MAC broadcast of the client so that not all DHCP relay devices receive the same request from the client duplicated. Without this option in place, as all client requests are broadcast packets in a VLAN, the original request gets duplicated through VXLAN and sent to all fabric nodes which have that VRF configured, which then perform DHCP relay towards the WAN router.

```
set interfaces irb unit <vlan-id> no-dhcp-flood
```

NOTE: Should you use the [vJunos-switch](#) as a virtual switch instance in a lab, it is known that while the “no-dhcp-flood” option can be configured on a virtual switch, it is not currently implemented. Hence, you may observe flooding and the wrong gateway IP address used. However, that should not affect your ability to test this. It's just a known limitation of the vJunos-switch.

Validation Framework

IN THIS SECTION

- [Test Bed | 15](#)
- [Platforms / Devices Under Test \(DUT\) | 17](#)
- [Test Bed Configuration | 17](#)

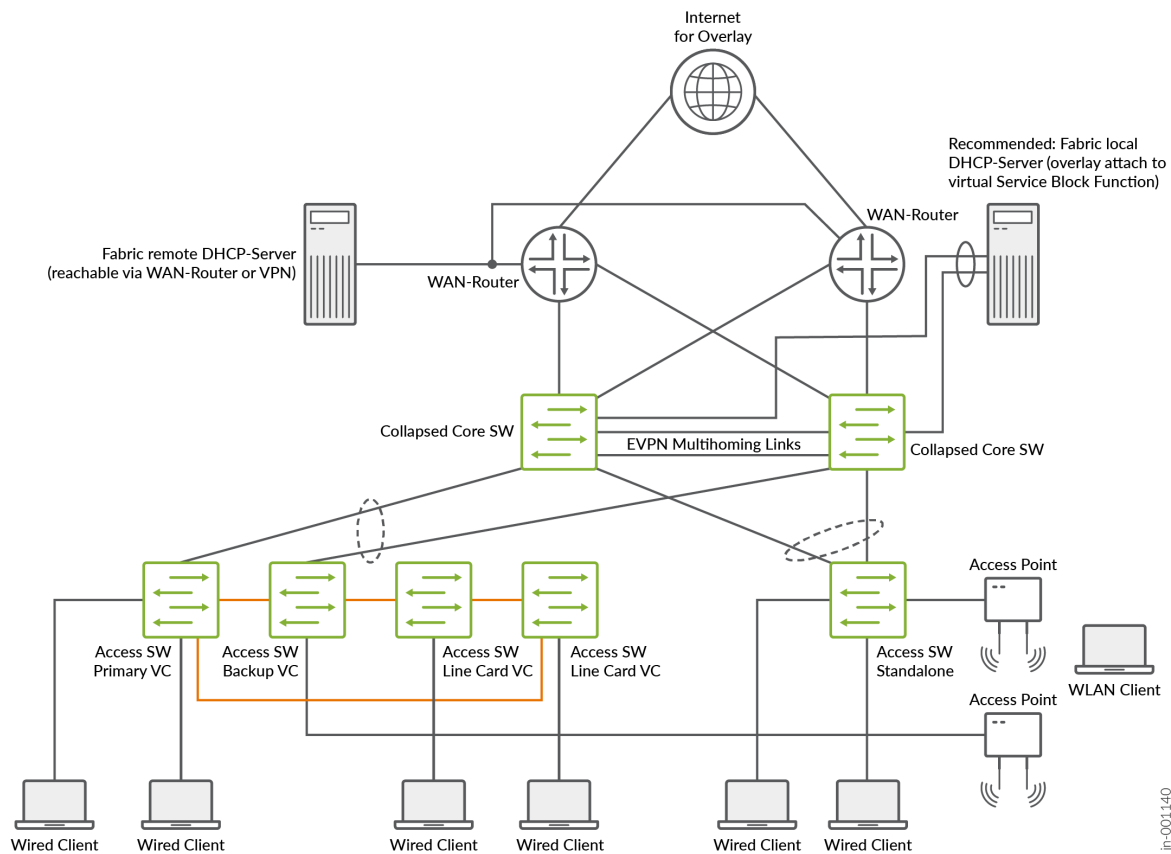
Test Bed

We are going to test two different fabric types which results in two different network topologies being used for testing.

The first one is EVPN multihoming as a virtual gateway fabric where the DHCP relay gateway IP address embedded is based on the static IP address the VRF got assigned per VRF:

- In the figure below, you can see the deployment of the DHCP server happens locally as part of the fabric as ESI-LAG integration (the service-block is a virtual function on the collapsed core in this scenario).
- Also, the other form of integration using a remote DHCP server not directly attached to the fabric can be tested in this design.

Figure 3: EVPN Multihoming with Two Collapsed Cores



The second is a 5-stage IP Clos fabric as an anycast fabric where the DHCP relay gateway IP address used will be one taken from the overlay loopback IP address pool. Remember those are /32 addresses unique to a VRF assigned on a node:

- In the figure below, you can see the deployment of the DHCP server happens locally as part of the fabric as ESI-LAG integration (the service block is comprised of separate physical hardware in this scenario).
- Also, the other form of integration using a remote DHCP server not directly attached to the fabric is shown in the figure.
- Two or more PoDs are optional.
- A Juniper Mist Edge deployment with Wi-Fi client is broken out of the fabric at the service block function and is emulated through a single attached wired client.

Test Objectives

IN THIS SECTION

- [Test Goals | 18](#)
- [Test Non-Goals | 19](#)

Test Goals

The testing for this JVD was performed with the following goals in mind:

- This solution was tested with two major fabric types:
 - EVPN multihomed fabric with a Layer 2 transport VLAN attached to WAN router. The approach for this fabric is to use static IPs from VGA as relay gateway IPs.
 - IP Clos 5-stage fabric with BGP protocol attached to WAN router. Physical service blocks are optional but recommended. The approach for this fabric is to use extra loopback IPs as relay gateway IPs.
- The two major DHCP server integration types must be tested with each of the two fabric types:
 - Local attach to the fabric through the service block function:
 - Test the DHCP server as part of the same VRF which does not involve the WAN router.
 - Test the DHCP server in a different VRF which does involve the WAN router for hair-pinning of traffic.
 - Remote attachment of the DHCP server outside of the fabric. In this case, you must ensure that no form of NAT between the fabric WAN router and the DHCP server is performed.
- All tests with the two fabric and integration types must happen with the two major DHCP server flavours we test for this solution from different third-party vendors:
 - Microsoft 2022 DHCP Server (as VM).
 - Linux KEA DHCP server (as VM).
- A client must get a DHCP lease as:
 - New MAC address with the full cycle of (Discover, Offer, Request and Ack).

- Lease renew (Request and Ack only).
- Special client situations must be tested such as:
 - Getting a DHCP release from clients attached to a service block function. This will emulate the behaviour of a Juniper Mist Edge device integrated into the fabric managing Wi-Fi clients at scale.
 - Wi-Fi roaming to a new VLAN. The MAC address of the client needs to be the same but when it connects to a new port, a different VLAN then needs to be assigned. This will simulate a roaming Wi-Fi client between buildings and SSIDs.

NOTE: Review the separate Test Report for more detailed information.

Test Non-Goals

The testing for this JVD was not performed, for various reasons, on the following items:

- Testing with many different third-party DHCP servers. Our testing focuses on two major products seen in the field:
 - Microsoft DHCP server in Windows 2022 as a commercial product.
 - Linux-based KEA DHCP server as free and professional alternative from the Internet Systems Consortium.
- Redundancy of the DHCP server itself will not be tested. An internal lease sharing Database and how to set it up is subject to the customer DHCP server installation.
- Testing with CRB or ERB fabric designs was not performed. With the tests on EVPN multihoming and IP Clos, the major differences are captured already.
- Testing with two integrated WAN routers. This was already tested as part of the [JVDE for WAN router integration](#).
- Microsoft DHCP server in Windows 2022 was not integrated into an AD, nor was it responsible for DNS management. This is subject to individual customer integration.

Recommendations

The following simple guidelines will help you to successfully implement DHCP relay using campus fabric.

Like WAN router integration, at the start of the campus fabric integration, the capabilities of the third-party DHCP server provided by the customer should be verified to meet the following requirements:

- It must provide at least two DHCP server instances for redundancy of this critical function. State sharing of the DHCP lease database between these instances simplifies the design by not needing to configure non-overlapping IP address ranges among separate instances.
- For DHCP relay to work properly, the DHCP server is required to listen on an IP address-based socket-interface.
- The location of the DHCP server is an important consideration. The closer to clients, the better. Attaching the DHCP server to the fabric should be considered.
- The DHCP server should be able to analyse the embedded option 82 information to make the decision on the DHCP lease assignment. Without additional Junos OS CLI configuration, the fabric will send:
 - Option 82 sub-option 1 (circuit ID) with the originating VLAN ID as a string.
 - Option 82 sub-option 5+11 contains the 4 bytes of the originating subnet.
 - The gateway IP address can also be used when the fabric is a virtual gateway fabric type.
- DNS integration with security systems (such as NAC) for monitoring client state should be left to the customer maintaining the third-party DHCP server.

Here, we summarize the best practice items already shared in the solution architecture section:

- Be aware that the communication towards the DHCP server happens based on embedded gateway IP addresses, which is dependent on the fabric type:
 - Static IP address of the overlay VLANs is used in virtual gateway fabrics such as:
 - In EVPN multihoming, this is always the case.
 - In CRB, this is optional when deleting the overlay loopback pool prefix from the fabric configuration.
 - Overlay loopback IP addresses coming from a pool and assigned globally as /32 across all VRF's:
 - In CRB, this is optional when configuring the overlay loopback pool prefix from the fabric configuration.
 - In ERB, this is recommended, so configure the overlay loopback pool prefix in the fabric configuration.
 - In IP Clos, this is recommended, so configure the overlay loopback pool prefix in the fabric configuration.

- When using overlay loopback IP addresses, ensure to use a routing protocol such as eBGP or OSPF towards the WAN router as it makes the propagation of the distributed /32 IP addresses over the VRFs easier to maintain.
- If your fabric was deployed before June 21, 2024, you may need additional Junos OS configuration to propagate the /32 overlay loopback IP addresses. Please review the appendix for an example.
- The location of the DHCP server needs to be validated:
 - We recommend directly integrating the DHCP server into the service block function of the fabric.
 - Less preferred is locating the DHCP server outside the fabric. If this must be done, please make sure that:
 - You minimize the latency between the fabric and DHCP server. Deploying the DHCP server in a public cloud should be avoided, if possible.
 - No form of network address (NAT) translation between the fabric and the DHCP server is performed.
- All DHCP relay forwarding configuration must be performed as part of the campus fabric configuration dialogue when configuring networks and VRFs. Do not override the fabric DHCP relay configuration by manually configuring it on the switch.

For debugging, we also recommend the following practices:

- Deploy a spare Juniper access point in the fabric that you are allowed to reboot anytime during the DHCP server integration. The DHCP client on the access point is perfect for triggering a DHCP lease request for testing and you will directly see the effects of the lease assignment in the Juniper Mist portal.
- When activating DHCP snooping:
 - This is only allowed to be performed on the access switches in the fabric where DHCP traffic ingresses.
 - Turn this function off if you do not need it any longer. In contrast to a typical EX series switch branch deployment where this function can be used to report the IP address clients, fabrics usually have a VRF where the IP address for a given MAC address is reported to the Juniper Mist cloud. This also happens for static assigned IP addresses of clients in the fabric which is the reason it's not needed really.

APPENDIX: Example Microsoft DHCP Server Configuration Used for Testing

Use Microsoft PowerShell CLI to implement the DHCP server configuration for the three example VLANs used for testing:

```
# enable Ping to DHCP server for connection debugging
netsh advfirewall firewall add rule name="Allow ICMPv4" protocol=icmpv4:8,any dir=in action=allow
Install-WindowsFeature DHCP -IncludeManagementTools
netsh dhcp add securitygroups
Restart-Service dhcpserver
# When using AD you must add the DHCPs server to the AD by it's static ipaddress
# Add-DhcpServerInDC -DnsName DHCP1.example.com -IPAddress 192.168.122.11
# Get-DhcpServerInDC
Add-DhcpServerv4Scope -Name "VLAN1033" -StartRange 10.33.33.10 -EndRange 10.33.33.250 -
SubnetMask 255.255.255.0
Set-DhcpServerv4OptionValue -ScopeId 10.33.33.0 -OptionId 3 -Value "10.33.33.1"
Set-DhcpServerv4OptionValue -ScopeId 10.33.33.0 -OptionId 6 -Value "8.8.8.8", "9.9.9.9"
Add-DhcpServerv4Scope -Name "VLAN1088" -StartRange 10.88.88.10 -EndRange 10.88.88.250 -
SubnetMask 255.255.255.0
Set-DhcpServerv4OptionValue -ScopeId 10.88.88.0 -OptionId 3 -Value "10.88.88.1"
Set-DhcpServerv4OptionValue -ScopeId 10.88.88.0 -OptionId 6 -Value "8.8.8.8", "9.9.9.9"
Add-DhcpServerv4Scope -Name "VLAN1099" -StartRange 10.99.99.10 -EndRange 10.99.99.250 -
SubnetMask 255.255.255.0
Set-DhcpServerv4OptionValue -ScopeId 10.99.99.0 -OptionId 3 -Value "10.99.99.1"
Set-DhcpServerv4OptionValue -ScopeId 10.99.99.0 -OptionId 6 -Value "8.8.8.8", "9.9.9.9"
# define Fabric loopback scope that is disabled to prevent lease handout
# and using what is reported via junos "relay-option-82 server-id override"
Add-DhcpServerv4Scope -Name "Fabric LoopbackIPs" -StartRange 172.16.192.1 -EndRange
172.16.223.254 -SubnetMask 255.255.224.0 -State InActive
# allow DHCP-clients to change previously from assigned VLAN to new VLAN
# (helps in Wi-Fi Roaming to new buildings/SSID's)
Set-ItemProperty -Path "HKLM:\System\CurrentControlSet\Services\DhcpServer\Parameters" -name
"DhcpFlagSubnetChangeDHCPRequest" -value 1
```


APPENDIX: Example Linux KEA DHCP Server Configuration Used for Testing

We've used an Ubuntu 22.04 VM as we needed KEA DHCP to listen to socket-interfaces, which is not available in versions of KEA prior to 2.0. Make sure the KEA DHCP server is on version 2.0 or higher.

The configuration to determine the original VLAN is based on the "client-classes" configuration. There, we analyse the option 82 sub-option 1 field containing the original VLAN ID and map that to a "name" attribute. This attribute then must be referenced as a "client-class" attribute in the "subnet4" configuration for each VLAN. In our example, the attribute parsing happens as hex values of the original VLAN ID string example:

- VLAN ID = "1099" sent as "option[82].option[1]" field string.
- The ASCII values of each character in decimal are 49+48+57+57.
- The ASCII values of each character in hex are 31+30+39+39 hence you parse for 0x31303939.

```
apt-get install -y kea net-tools bridge-utils
kea-shell -v
2.0.2
cat <<EOF >/etc/kea/kea-api-password
juniper123
EOF
chmod 0640 /etc/kea/kea-api-password
chown root /etc/kea/kea-api-password
chgrp _kea /etc/kea/kea-api-password
ls -l /etc/kea/kea-api-password
dpkg-reconfigure kea-ctrl-agent
cp /etc/kea/kea-dhcp4.conf /etc/kea/kea-dhcp4.conf.orig
cat <<EOF >/etc/kea/kea-dhcp4.conf
{
  "Dhcp4": {
    "interfaces-config": {
      "interfaces": [ "*" ],
      "dhcp-socket-type": "udp"
    },
    "control-socket": {
      "socket-type": "unix",
      "socket-name": "/tmp/kea4-ctrl-socket"
    }
  },
```

```

"lease-database": {
  "type": "memfile",
  "lfc-interval": 3600
},
"expired-leases-processing": {
  "reclaim-timer-wait-time": 10,
  "flush-reclaimed-timer-wait-time": 25,
  "hold-reclaimed-time": 3600,
  "max-reclaim-leases": 100,
  "max-reclaim-time": 250,
  "unwarned-reclaim-cycles": 5
},
"renew-timer": 900,
"rebind-timer": 1800,
"valid-lifetime": 3600,
"option-data": [
  {
    "name": "domain-name-servers",
    "data": "8.8.8.8, 9.9.9.9"
  }
],
"client-classes": [
  {
    "name": "vlan1099",
    "test": "option[82].option[1].hex == 0x31303939"
  },
  {
    "name": "vlan1088",
    "test": "option[82].option[1].hex == 0x31303838"
  },
  {
    "name": "vlan1033",
    "test": "option[82].option[1].hex == 0x31303333"
  }
],
"subnet4": [
  {
    "id": 1,
    "subnet": "10.99.99.0/24",
    "pools": [
      {
        "pool": "10.99.99.10 - 10.99.99.250"
      }
    ]
  }
]

```

```

    ],
    "option-data": [
      {
        "name": "routers",
        "data": "10.99.99.1"
      }
    ],
    "client-class": "vlan1099"
  },
  {
    "id": 2,
    "subnet": "10.88.88.0/24",
    "pools": [
      {
        "pool": "10.88.88.10 - 10.88.88.250"
      }
    ],
    "option-data": [
      {
        "name": "routers",
        "data": "10.88.88.1"
      }
    ],
    "client-class": "vlan1088"
  },
  {
    "id": 3,
    "subnet": "10.33.33.0/24",
    "pools": [
      {
        "pool": "10.33.33.10 - 10.33.33.250"
      }
    ],
    "option-data": [
      {
        "name": "routers",
        "data": "10.33.33.1"
      }
    ],
    "client-class": "vlan1033"
  }
],
"loggers": [

```

```

{
    "name": "kea-dhcp4",
    "output_options": [
        {
            "output": "syslog"
        }
    ],
    "severity": "DEBUG",
    "debuglevel": 0
}
]
}
}
EOF
systemctl restart kea-dhcp4-server
tail -f /var/log/syslog
systemctl status kea-dhcp4-server
* kea-dhcp4-server.service - Kea IPv4 DHCP daemon
   Loaded: loaded (/lib/systemd/system/kea-dhcp4-server.service; enabled; vendor preset:
   enabled)
   Active: active (running) since Mon 2024-04-22 14:30:39 UTC; 1min 33s ago
     Docs: man:kea-dhcp4(8)
  Main PID: 2342 (kea-dhcp4)
    Tasks: 5 (limit: 2220)
   Memory: 2.6M
      CPU: 17ms
   CGroup: /system.slice/kea-dhcp4-server.service
           └─2342 /usr/sbin/kea-dhcp4 -c /etc/kea/kea-dhcp4.conf

netstat -tunap
Active Internet connections (servers and established)

```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program
tcp	0	0	127.0.0.1:8000	0.0.0.0:*	LISTEN	1598/kea-ctrl-agent
udp	0	0	192.168.122.12:67	0.0.0.0:*		2083/kea-dhcp4

APPENDIX: Example DHCP Relay in EVPN Multihoming Fabric

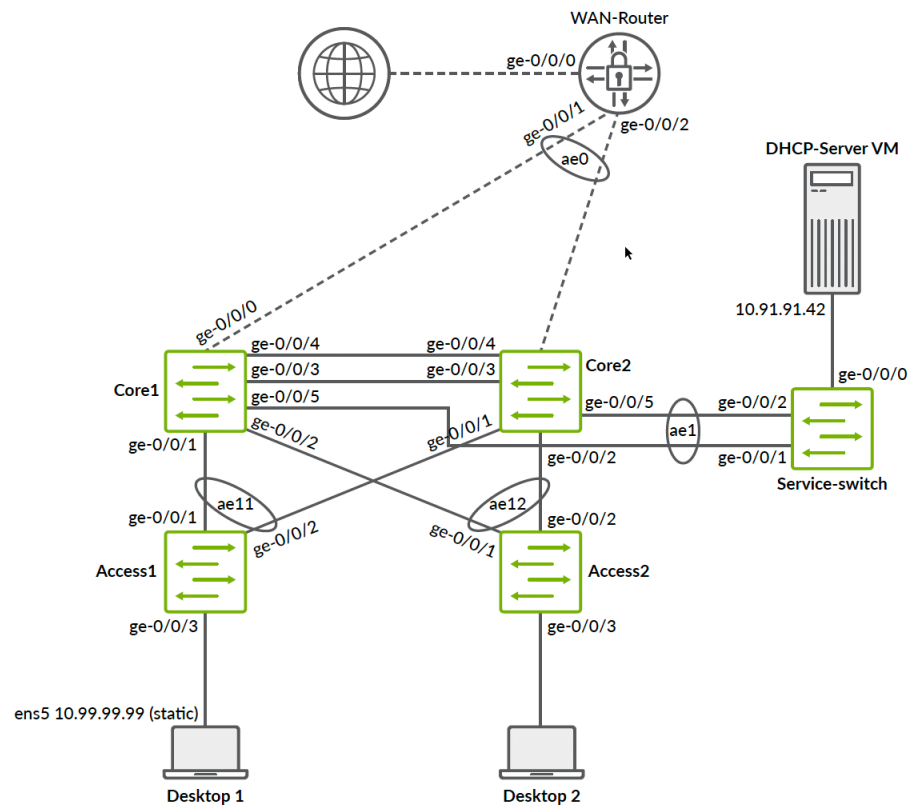
IN THIS SECTION

- [Campus Fabric Dialogue Configuration | 32](#)
- [Access Switch Configuration | 35](#)
- [DHCP Server Integration | 37](#)
- [Check the DHCP Server Itself | 40](#)
- [Final Test with a Wired Client | 42](#)
- [Example Trace of a DHCP Lease Handout | 45](#)

Following is an example lab design to test DHCP relay in an EVPN multihoming virtual gateway fabric with the following configuration:

- Fabric type = EVPN multihoming.
- Overlay loopback pool configured = N/A (static IRB IP addresses used as gateway IP addresses)
- WAN router integration = was not performed yet at this point of testing.
- DHCP server location = Integrated with the fabric through a virtual service block function.
- DHCP server reachability = Inside the fabric between different VLANs assigned to the same VRF.
- Third-party DHCP server used = Microsoft Windows 2022 DHCP server as VM.

NOTE: The additional server switch was only added because when Microsoft Windows 2022 is installed in a VM, LACP bundling is not available for “team” interfaces. Hence, this was the better option to translate between these two worlds.



NOTE: The configuration example shown below is only showing configuration relevant to the DHCP relay integration to make it brief. The full workflow can be deduced from available JVDs and extensions posted already.

You can reuse the switch template we have provided by importing the JSON below to expedite progress with this configuration:

```
{
  "ntp_servers": [],
  "dns_servers": [
    "8.8.8.8",
    "9.9.9.9"
  ],
  "dns_suffix": [],
  "additional_config_cmds": [],
  "networks": {
    "vlan1099": {
```

```

        "vlan_id": 1099,
        "subnet": "10.99.99.0/24"
    },
    "vlan1091": {
        "vlan_id": "1091",
        "subnet": "10.91.91.0/24",
        "subnet6": ""
    }
},
"port_usages": {
    "vlan1099": {
        "mode": "access",
        "disabled": false,
        "port_network": "vlan1099",
        "voip_network": null,
        "stp_edge": false,
        "all_networks": false,
        "networks": null,
        "port_auth": null,
        "speed": "auto",
        "duplex": "auto",
        "mac_limit": 0,
        "persist_mac": false,
        "poe_disabled": false,
        "enable_qos": false,
        "storm_control": {},
        "mtu": 9014,
        "description": "",
        "disable_autoneg": false
    },
    "vlan1091": {
        "disabled": false,
        "mode": "access",
        "port_network": "vlan1091",
        "voip_network": null,
        "stp_edge": false,
        "stp_p2p": false,
        "stp_no_root_port": false,
        "mac_auth_protocol": null,
        "all_networks": false,
        "networks": null,
        "port_auth": null,
        "allow_multiple_suplicants": null,

```

```

    "enable_mac_auth": null,
    "mac_auth_only": null,
    "guest_network": null,
    "bypass_auth_when_server_down": null,
    "dynamic_vlan_networks": null,
    "speed": "auto",
    "duplex": "auto",
    "mac_limit": 0,
    "persist_mac": false,
    "poe_disabled": false,
    "enable_qos": false,
    "storm_control": {},
    "mtu": 9014,
    "description": "",
    "disable_autoneg": false
  },
  "dynamic": {
    "mode": "dynamic",
    "reset_default_when": "link_down",
    "rules": []
  }
},
"switch_matching": {
  "enable": true,
  "rules": [
    {
      "name": "core",
      "port_config": {
        "ge-0/0/5": {
          "usage": "vlan1091",
          "dynamic_usage": null,
          "critical": false,
          "description": "",
          "no_local_overwrite": true,
          "aggregated": true,
          "ae_disable_lacp": false,
          "ae_lacp_force_up": false,
          "ae_lacp_slow": false,
          "ae_idx": 1,
          "esilag": true
        }
      }
    }
  ],
  "additional_config_cmds": [

```



```

        "set groups top routing-instances customera forwarding-options dhcp-relay forward-
only",
        "set groups top routing-instances customera forwarding-options dhcp-relay group
vlan1099 relay-option-82 circuit-id vlan-id-only",
        "set groups top routing-instances customera forwarding-options dhcp-relay group
vlan1099 relay-option-82 server-id-override",
        "",
        ""
    ],
    "oob_ip_config": {
        "type": "dhcp",
        "use_mgmt_vrf": false
    },
    "ip_config": {
        "type": "dhcp",
        "network": "default"
    },
    "port_mirroring": {},
    "match_name[0:4]": "core"
}
]
},
"switch_mgmt": {
    "config_revert_timer": 10,
    "root_password": "juniper123",
    "protect_re": {
        "enabled": false
    },
    "tacacs": {
        "enabled": false
    }
},
"radius_config": {
    "auth_servers": [],
    "acct_servers": [],
    "auth_servers_timeout": 5,
    "auth_servers_retries": 3,
    "fast_dot1x_timers": false,
    "acct_interim_interval": 0,
    "auth_server_selection": "ordered",
    "coa_enabled": false,
    "coa_port": ""
},

```

```

"vrf_config": {
  "enabled": false
},
"remote_syslog": {
  "enabled": false
},
"snmp_config": {
  "enabled": false
},
"dhcp_snooping": {
  "enabled": false
},
"acl_policies": [],
"mist_nac": {
  "enabled": true,
  "network": null
},
"disabled_system_defined_port_usages": [],
"bgp_config": null,
"routing_policies": {},
"port_mirroring": {},
"name": "DHCP Lab1"
}

```

Campus Fabric Dialogue Configuration


In the campus fabric dialogue configuration, it is important to configure:

- The correct fabric type = EVPN multihoming
- Virtual gateway v4 MAC address = Enabled (this makes debugging easier as there is not a global MAC address used across all VLANs).


Choose Campus Fabric Topology

Choose the topology you want to construct and configure related options


TOPOLOGY TYPE



EVPN Multihoming
Collapsed core with ESI-Lag



Campus Fabric Core-Distribution
EVPN core/distribution with ESI-Lag



Campus Fabric IP Clos
Campus fabric with L3 at the edge

CONFIGURATION

Topology Name

Virtual Gateway v4 MAC Address

Virtual gateway MAC auto-generated per network on the L3 gateway

☒ Enabled ☐ Disabled

OVERLAY SETTINGS

BGP Local AS

(2-byte or 4-byte)

UNDERLAY SETTINGS

AS Base

(2-byte or 4-byte)

Underlay

☒ IPv4 ☐ IPv6

Subnet ⓘ

(xxx.xxx.xxx.xxx/xx)

Auto Router ID Subnet / Loopback Interface ⓘ

(xxx.xxx.xxx.xxx/xx)

Assign the switches as collapsed cores and access switches in the next pane.

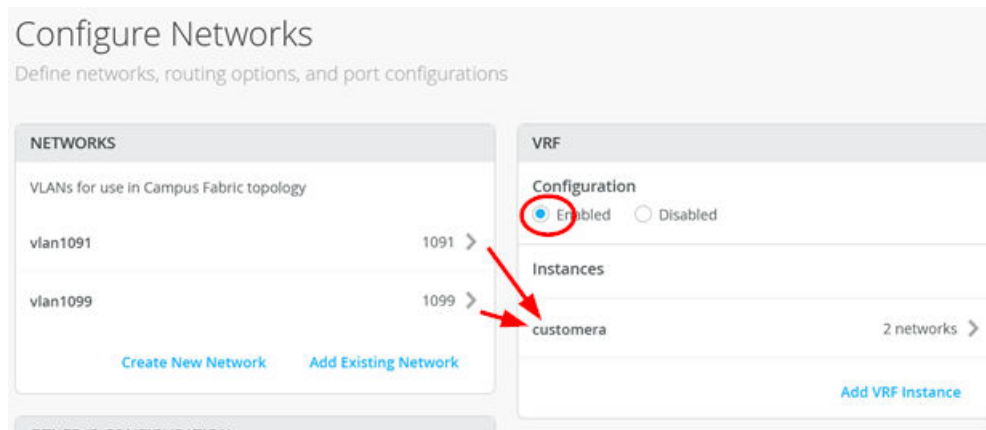
Then, in the “Networks” fabric dialogue, import two VLANs from the switch template.

- First VLAN (your desktop client requiring a lease will be here):
 - Name = vlan1099
 - VLAN ID = 1099
 - IPv4 subnet = 10.99.99.0/24
 - IPv4 virtual gateway = 10.99.99.1
- Second VLAN (your DHCP server will be here):
 - Name = vlan1091
 - VLAN ID = 1091
 - IPv4 subnet = 10.91.91.0/24
 - IPv4 virtual gateway = 10.91.91.1

Then, add them to a single VRF:

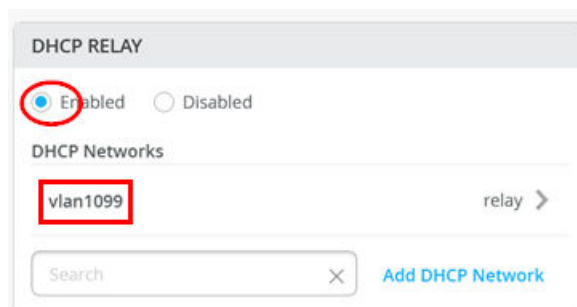
- VRF Configuration = Enabled
- VRF:
 - Name = customera
 - Networks = vlan1099 and vlan1091

See the example:



Also add the DHCP relay information:

- DHCP relay = Enabled
- vlan1099:
 - Network = vlan1099
 - DHCP servers = 10.91.91.42



DHCP RELAY

Edit DHCP Network

Network
vlan1099 1099

DHCP Servers
10.91.91.42
(Comma-separated IPs(v4 or v6))

NOTE: Ensure you always use this dialogue to configure DHCP relay in all campus fabric designs.

Finish the remaining dialogues so that Juniper Mist cloud pushes configuration to setup the fabric.

Access Switch Configuration

Our Desktop1 client is attached to the ge-0/0/3 interface on the Access1 switch through the following port configuration:

PORT CONFIGURATION

Port Profile Assignment
★ Site, Template, or System Defined

Edit Port Configuration

Port IDs
ge-0/0/3
(ge-0/0/1, ge-0/0/4, ge-0/1/1-23, etc)

Interface
☒ L2 interface
 ☐ L3 interface
 ☐ L3 sub-interfaces

Configuration Profile
 vlan1099 vlan1099(1099), access

☐ Enable Dynamic Port Configuration

Based on this configuration, Juniper Mist cloud configures the following Junos OS configuration on the two collapsed core switches where the VRF is located in this fabric.

```
set groups top routing-instances customera instance-type vrf
set groups top routing-instances customera interface irb.1091
set groups top routing-instances customera interface irb.1099
set groups top routing-instances customera forwarding-options dhcp-relay server-group vlan1099
10.91.91.42
      set groups top routing-instances customera forwarding-options dhcp-relay group
vlan1099 interface irb.1099
      set groups top routing-instances customera forwarding-options dhcp-relay group
vlan1099 active-server-group vlan1099
set groups top routing-instances customera route-distinguisher 172.16.254.1:101
set groups top routing-instances customera vrf-target target:65000:101
set groups top routing-instances customera vrf-table-label
set groups top routing-instances customera routing-options auto-export
```

NOTE: At the time this document was created, not all options in Junos OS that are explained in the solution architecture chapter are created. Hence, we manually added the below additional Junos OS configuration on each collapsed core switch to have the optimal experience with the various DHCP servers tested.

Additional CLI Commands ⓘ

```
set groups top routing-instances customera forwarding-options dhcp-relay forward-only
set groups top routing-instances customera forwarding-options dhcp-relay group vlan1099 relay-option-82 circuit-id vlan-id-only
set groups top routing-instances customera forwarding-options dhcp-relay group vlan1099 relay-option-82 server-id-override
```

```
set groups top routing-instances customera forwarding-options dhcp-relay forward-only
set groups top routing-instances customera forwarding-options dhcp-relay group vlan1099 relay-
option-82 circuit-id vlan-id-only
set groups top routing-instances customera forwarding-options dhcp-relay group vlan1099 relay-
option-82 server-id-override
```

DHCP Server Integration

We decided to position a switch between the fabric and the DHCP server which was a virtual machine. Microsoft Windows does not support the usage of LACP when the server is a VM. However, we do that at the fabric side, hence we use a switch as a kind of translation element. Another advantage is that towards the VM, we then have a single interface containing all messages which makes it easier to debug.

On the two collapsed core switches you configure:

- Port ID = ge-0/0/5
- Configuration profile = vlan1091
- Port aggregation = Enabled
 - AE Index = 1
 - ESI-LAG = Enabled

Edit Port Configuration ✓ ✕

Please ensure AE index does not overlap across different ports between Device, Site/Template and Campus Fabric configuration

Port IDs

ge-0/0/5

(ge-0/0/1, ge-0/0/4, ge-0/1/1-23, etc)

Configuration Profile

vlan1091 vlan1091(1091), access ✓

☐ Enable Dynamic Port Configuration

Description

Add Description

Up / Down Port Alerts ⓘ

☐ Enabled ☒ Disabled

Manage Alert Types in [Alerts Page](#)

Port Aggregation

☒ Enabled ☐ Disabled

LACP

☒ Enabled ☐ Disabled

LACP Force-UP ⓘ

☐ Enabled ☒ Disabled

LACP Periodic Slow

☐ Enabled ☒ Disabled

AE Index 1 (0 - 255)

☒ ESI-LAG

Allow switch port operator to modify port profile

☐ Yes ☒ No

On the switch between the fabric and the DHCP server, you add a simple LAG configuration such as the one below:

- Port IDs = ge-0/0/1-2
- Interface = L2 interface
- Configuration profile = vlan1091

- Port aggregation = Enabled
 - LACP = Enabled
 - LACP force up = Disabled
 - LACP periodic slow = Disabled
- AE Index = 1

The screenshot shows the 'Edit Port Configuration' window with the following settings:

- Port IDs:** `ge-0/0/1-2` (circled in red)
- Interface:** ☒ L2 interface (circled in red), ☐ L3 interface, ☐ L3 sub-interfaces
- Configuration Profile:** `vlan1091` (circled in red), `vlan1091(1091), access`
- Enable Dynamic Port Configuration:** ☐
- Description:** Add Description
- Up / Down Port Alerts:** ☐ Enabled, ☒ Disabled
- Port Aggregation:** ☒ Enabled (circled in red), ☐ Disabled
- LACP:** ☒ Enabled (circled in red), ☐ Disabled
- LACP Force-UP:** ☐ Enabled, ☒ Disabled (circled in red)
- LACP Periodic Slow:** ☐ Enabled, ☒ Disabled (circled in red)
- AE Index:** `1` (circled in red), (0 - 255)

The single interface towards the DHCP server then:

- Port ID = `ge-0/0/0`

- Interface = L2 interface
- Configuration profile = vlan1091

Edit Port Configuration

Port IDs
ge-0/0/0
(ge-0/0/1, ge-0/0/4, ge-0/1/1-23, etc)

Interface
☒ L2 interface
 ☐ L3 interface
 ☐ L3 sub-interfaces

Configuration Profile
 vlan1091
 vlan1091(1091), access ▼

☐ Enable Dynamic Port Configuration

Check the DHCP Server Itself

We assume that you have installed Microsoft Windows 2022 Server edition and applied the configuration example from above chapter ["APPENDIX: Example Microsoft DHCP server configuration used for testing" on page 22](#) . When this is done, you need to check the connection to the fabric to ensure the server gets the relay messages. In our case, the server had more than one interface for bootstrapping it:

```
PS C:\Users\Administrator> ipconfig
Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::7d5a:47ae:2892:3d88%5
    IPv4 Address. . . . . : 192.168.10.200
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 

Ethernet adapter Ethernet 2:

    Connection-specific DNS Suffix  . : 
    IPv4 Address. . . . . : 10.91.91.42
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 10.91.91.1

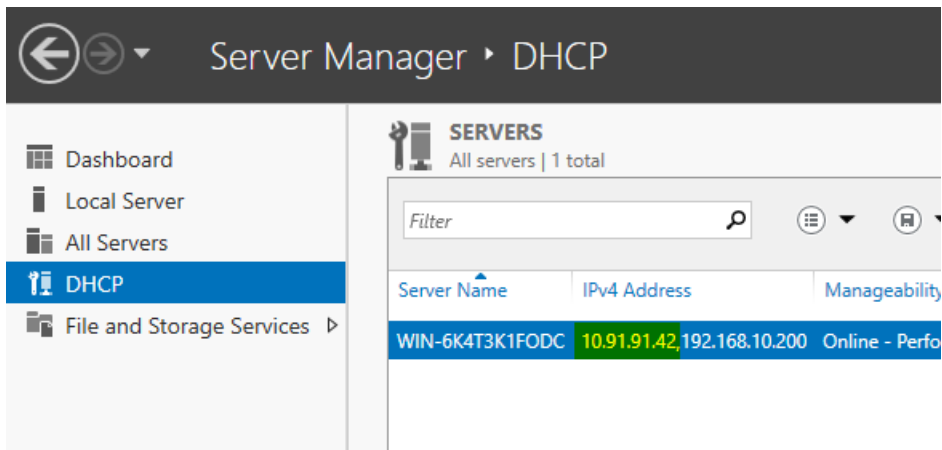
# ping the VGA of this subnet
PS C:\Users\Administrator> ping 10.91.91.1
```

```

Pinging 10.91.91.1 with 32 bytes of data:
Reply from 10.91.91.1: bytes=32 time=1ms TTL=64
Reply from 10.91.91.1: bytes=32 time=1ms TTL=64
Reply from 10.91.91.1: bytes=32 time=1ms TTL=64
Reply from 10.91.91.1: bytes=32 time=1ms TTL=64
Ping statistics for 10.91.91.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms
# ping desktop1 client who should have a static IP-Address now
PS C:\Users\Administrator> ping 10.99.99.99
Pinging 10.99.99.99 with 32 bytes of data:
Reply from 10.99.99.99: bytes=32 time=2ms TTL=63
Reply from 10.99.99.99: bytes=32 time=2ms TTL=63
Reply from 10.99.99.99: bytes=32 time=2ms TTL=63
Reply from 10.99.99.99: bytes=32 time=1ms TTL=63
Ping statistics for 10.99.99.99:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 2ms, Average = 1ms

```

Verify that the server is listening on the IP address 10.91.91.42:



Final Test with a Wired Client

To complete the installation, perform a final test with a wired client. The initial state of our client is that it has a static IP address assigned and can communicate towards the internet:

```
root@desktop1:~# ifconfig ens5
ens5: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.99.99.99 netmask 255.255.255.0 broadcast 10.99.99.255
    inet6 fe80::5054:ff:feae:1e5b prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:ae:1e:5b txqueuelen 1000 (Ethernet)
    RX packets 142 bytes 10653 (10.6 KB)
    RX errors 0 dropped 51 overruns 0 frame 0
    TX packets 158 bytes 14096 (14.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

# do ping the VGA shared GW IP
root@desktop1:~# ping -c3 10.99.99.1
PING 10.99.99.1 (10.99.99.1) 56(84) bytes of data.
64 bytes from 10.99.99.1: icmp_seq=1 ttl=64 time=0.785 ms
64 bytes from 10.99.99.1: icmp_seq=2 ttl=64 time=1.03 ms
64 bytes from 10.99.99.1: icmp_seq=3 ttl=64 time=1.19 ms
--- 10.99.99.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.785/1.003/1.193/0.167 ms
# we ping the static IP of the one collapsed core switch
root@desktop1:~# ping -c3 10.99.99.2
PING 10.99.99.2 (10.99.99.2) 56(84) bytes of data.
64 bytes from 10.99.99.2: icmp_seq=1 ttl=64 time=0.766 ms
64 bytes from 10.99.99.2: icmp_seq=2 ttl=64 time=0.959 ms
64 bytes from 10.99.99.2: icmp_seq=3 ttl=64 time=0.824 ms
--- 10.99.99.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.766/0.849/0.959/0.080 ms
# we ping the static IP of the other collapsed core switch
root@desktop1:~# ping -c3 10.99.99.3
PING 10.99.99.3 (10.99.99.3) 56(84) bytes of data.
64 bytes from 10.99.99.3: icmp_seq=1 ttl=64 time=2.91 ms
64 bytes from 10.99.99.3: icmp_seq=2 ttl=64 time=1.35 ms
64 bytes from 10.99.99.3: icmp_seq=3 ttl=64 time=1.43 ms
--- 10.99.99.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 1.354/1.897/2.912/0.718 ms
```

```
# we review the ARP-cache for the 3 MAC-Addresses
root@desktop1:~# arp -an
? (10.99.99.3) at 2c:6b:f5:f8:9b:f0 [ether] on ens5
? (10.99.99.2) at 2c:6b:f5:ac:da:f0 [ether] on ens5
? (10.99.99.1) at 00:00:5e:e4:04:4b [ether] on ens5
# we try to test connectivity towards DHCP server
# Server must allow ICMP Ping which is not a default
root@desktop1:~# ping -c3 10.91.91.42
PING 10.91.91.42 (10.91.91.42) 56(84) bytes of data.
64 bytes from 10.91.91.42: icmp_seq=1 ttl=127 time=2.35 ms
64 bytes from 10.91.91.42: icmp_seq=2 ttl=127 time=2.32 ms
64 bytes from 10.91.91.42: icmp_seq=3 ttl=127 time=2.51 ms
--- 10.91.91.42 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 2.322/2.392/2.506/0.081 ms
```

We can see this client and its IP address in the Wired Client overview in the Juniper Mist portal:

Client Name	MAC Address	VLAN	Wireless Clients	Switch	Port	Port Profile	IPv4 Address	Manufacturer	Insights
<> 52:54:00:ae:1e:5b	52:54:00:ae:1e:5b	1099	--	access1	ge-0/0/3	vlan1099	10.99.99.99	Unknown	Wired Client Insights
<> 52:54:00:73:46:1c	52:54:00:73:46:1c	1091	--	server	ge-0/0/0	vlan1091	10.91.91.42	Unknown	Wired Client Insights

Next, we unconfigure the static IP address and try to obtain a DHCP lease instead. The additional configuration below ensures that the client loses the static configuration and any prior knowledge of a DHCP lease. We then start up the DHCP client in the foreground to see a bit more of the debugging messages:

```
root@desktop1:~# ifconfig ens5 0.0.0.0 up
root@desktop1:~# pkill dhclient
root@desktop1:~# rm -f /var/lib/dhcp/*.leases
root@desktop1:~# dhclient -v ens5
Internet Systems Consortium DHCP Client 4.4.1
Copyright 2004-2018 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
Listening on LPF/ens5/52:54:00:ae:1e:5b
Sending on   LPF/ens5/52:54:00:ae:1e:5b
Sending on   Socket/fallback
DHCPDISCOVER on ens5 to 255.255.255.255 port 67 interval 3 (xid=0x8599fa19)
```

```

DHCP OFFER of 10.99.99.63 from 10.99.99.3
DHCP REQUEST for 10.99.99.63 on ens5 to 255.255.255.255 port 67 (xid=0x19fa9985)
DHCP ACK of 10.99.99.63 from 10.99.99.3 (xid=0x8599fa19)
bound to 10.99.99.63 -- renewal in 304223 seconds.

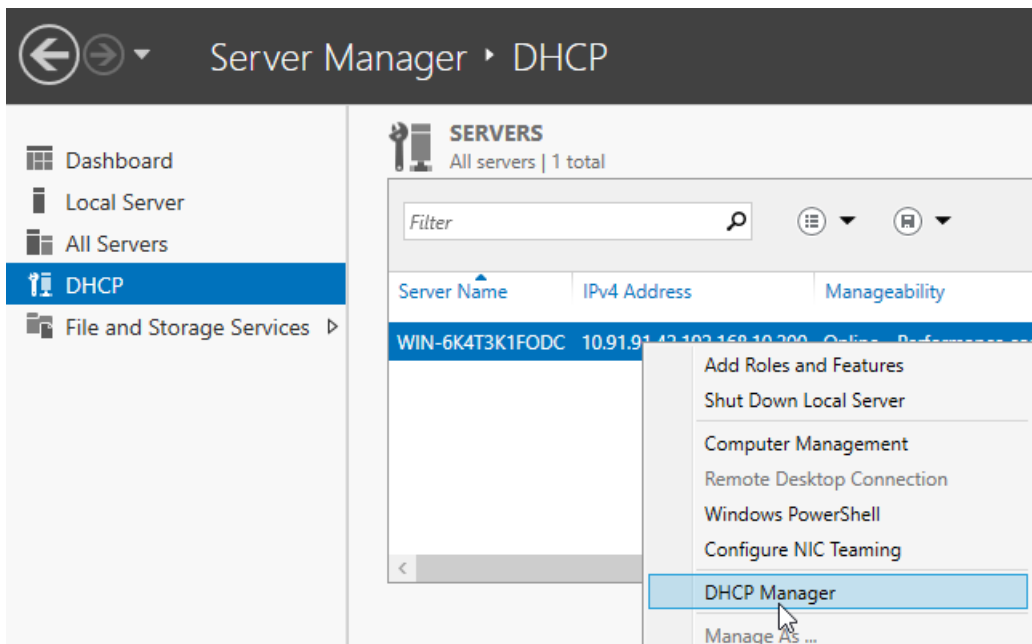
```

After a while (depending on local ARP age-outs), this change becomes visible in the Wired Client overview:

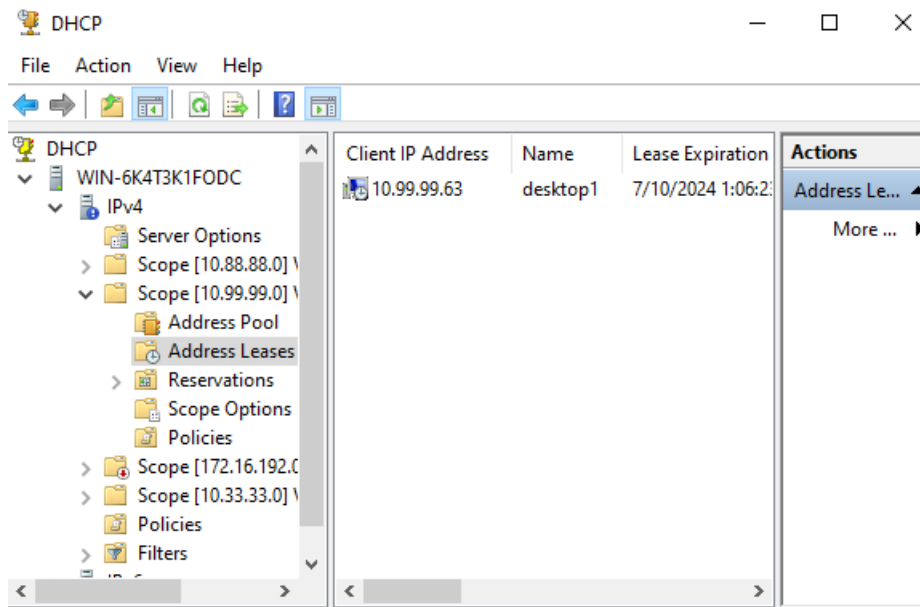


Client Name	MAC Address	VLAN	Wireless Clients	Switch	Port	Port Profile	IPv4 Address	Manufacturer	Insights
<> 52:54:00:ae:1e:5b	52:54:00:ae:1e:5b	1099	--	access1	ge-0/0/3	vlan1099	10.99.99.63	Unknown	Wired Client Insights
<> 52:54:00:73:46:1c	52:54:00:73:46:1c	1091	--	server	ge-0/0/0	vlan1091	10.91.91.42	Unknown	Wired Client Insights

You can also see the lease handout on the DHCP server itself when opening the DHCP Manager as follows:



Below is an example of our client now appearing in the lease database of the DHCP server:



Example Trace of a DHCP Lease Handout

Below, you find the full sequence of a DHCP lease handout looking at the originating wired client and the DHCP server side getting the forwarded messages from the relay agent of the fabric. This was captured simultaneously through tcpdump.

We start the trace on the client sending the discover message through broadcast to the fabric relay agent:

```
10:14:30.770921 52:54:00:ae:1e:5b > ff:ff:ff:ff:ff:ff, ethertype IPv4 (0x0800), length 342: (tos
0x10, ttl 128, id 0, offset 0, flags [none], proto UDP (17), length 328)
    0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 52:54:00:ae:1e:5b, length 300, xid
0xb7d35449, Flags [none]
        Client-Ethernet-Address 52:54:00:ae:1e:5b
        Vendor-rfc1048 Extensions
            Magic Cookie 0x63825363
            DHCP-Message Option 53, length 1: Discover
            Hostname Option 12, length 8: "desktop1"
            Parameter-Request Option 55, length 13:
                Subnet-Mask, BR, Time-Zone, Default-Gateway
                Domain-Name, Domain-Name-Server, Option 119, Hostname
                Netbios-Name-Server, Netbios-Scope, MTU, Classless-Static-Route
                NTP
```

We now review the relay message arriving on the DHCP server with the embedded information from the fabric relay agent such as gateway IP and option 82:

```
10:14:30.772605 2c:6b:f5:f8:9b:f0 > 52:54:00:73:46:1c, ethertype IPv4 (0x0800), length 394: (tos
0x0, ttl 64, id 19866, offset 0, flags [none], proto UDP (17), length 380)
    10.99.99.3.67 > 10.91.91.42.67: BOOTP/DHCP, Request from 52:54:00:ae:1e:5b, length 352, xid
0xb7d35449, Flags [none]
        Gateway-IP 10.99.99.3
        Client-Ethernet-Address 52:54:00:ae:1e:5b
        Vendor-rfc1048 Extensions
            Magic Cookie 0x63825363
            DHCP-Message Option 53, length 1: Discover
            Hostname Option 12, length 8: "desktop1"
            Parameter-Request Option 55, length 13:
                Subnet-Mask, BR, Time-Zone, Default-Gateway
                Domain-Name, Domain-Name-Server, Option 119, Hostname
                Netbios-Name-Server, Netbios-Scope, MTU, Classless-Static-Route
                NTP
            Agent-Information Option 82, length 50:
                Circuit-ID SubOption 1, length 4: 1099
                Unknown SubOption 5, length 4:
                    0x0000: 0a63 6303
                Unknown SubOption 11, length 4:
                    0x0000: 0a63 6303
                Unknown SubOption 9, length 30:
                    0x0000: 0000 0a4c 1904 1749 5242 2d69 7262 2e31
                    0x0010: 3039 393a 7674 6570 2e33 3237 3639
```

The DHCP server responds towards the fabric relay agent with the offer for the client:

```
10:14:30.773018 52:54:00:73:46:1c > 00:00:5e:e4:04:43, ethertype IPv4 (0x0800), length 384: (tos
0x0, ttl 128, id 47992, offset 0, flags [none], proto UDP (17), length 370)
    10.91.91.42.67 > 10.99.99.3.67: BOOTP/DHCP, Reply, length 342, xid 0xb7d35449, Flags [none]
        Your-IP 10.99.99.63
        Server-IP 10.91.91.42
        Gateway-IP 10.99.99.3
        Client-Ethernet-Address 52:54:00:ae:1e:5b
        Vendor-rfc1048 Extensions
            Magic Cookie 0x63825363
            DHCP-Message Option 53, length 1: Offer
            Subnet-Mask Option 1, length 4: 255.255.255.0
```



```

RN Option 58, length 4: 345600
RB Option 59, length 4: 604800
Lease-Time Option 51, length 4: 691200
Server-ID Option 54, length 4: 10.99.99.3
Default-Gateway Option 3, length 4: 10.99.99.1
Domain-Name-Server Option 6, length 8: 8.8.8.8,9.9.9.9
Agent-Information Option 82, length 50:
  Circuit-ID SubOption 1, length 4: 1099
  Unknown SubOption 5, length 4:
    0x0000: 0a63 6303
  Unknown SubOption 11, length 4:
    0x0000: 0a63 6303
  Unknown SubOption 9, length 30:
    0x0000: 0000 0a4c 1904 1749 5242 2d69 7262 2e31
    0x0010: 3039 393a 7674 6570 2e33 3237 3639

```

The fabric relay agent filters some of the information out (such as option 82) and forwards the offer as Layer 2 unicast to the client:

```

10:14:30.774479 2c:6b:f5:f8:9b:f0 > 52:54:00:ae:1e:5b, ethertype IPv4 (0x0800), length 332: (tos
0x0, ttl 64, id 19869, offset 0, flags [none], proto UDP (17), length 318)
  10.99.99.3.67 > 10.99.99.63.68: BOOTP/DHCP, Reply, length 290, xid 0xb7d35449, Flags [none]
    Your-IP 10.99.99.63
    Server-IP 10.91.91.42
    Client-Ethernet-Address 52:54:00:ae:1e:5b
    Vendor-rfc1048 Extensions
      Magic Cookie 0x63825363
      DHCP-Message Option 53, length 1: Offer
      Subnet-Mask Option 1, length 4: 255.255.255.0
      RN Option 58, length 4: 345600
      RB Option 59, length 4: 604800
      Lease-Time Option 51, length 4: 691200
      Server-ID Option 54, length 4: 10.99.99.3
      Default-Gateway Option 3, length 4: 10.99.99.1
      Domain-Name-Server Option 6, length 8: 8.8.8.8,9.9.9.9

```

Based on the previous offer received, the client now sends a request message through broadcast to the fabric relay agent:

```

10:14:30.776566 52:54:00:ae:1e:5b > ff:ff:ff:ff:ff:ff, ethertype IPv4 (0x0800), length 342: (tos
0x10, ttl 128, id 0, offset 0, flags [none], proto UDP (17), length 328)

```

```

0.0.0.0.68 > 255.255.255.255.67: BOOTP/DHCP, Request from 52:54:00:ae:1e:5b, length 300, xid
0xb7d35449, Flags [none]
  Client-Ethernet-Address 52:54:00:ae:1e:5b
  Vendor-rfc1048 Extensions
    Magic Cookie 0x63825363
    DHCP-Message Option 53, length 1: Request
    Server-ID Option 54, length 4: 10.99.99.3
    Requested-IP Option 50, length 4: 10.99.99.63
    Hostname Option 12, length 8: "desktop1"
    Parameter-Request Option 55, length 13:
      Subnet-Mask, BR, Time-Zone, Default-Gateway
      Domain-Name, Domain-Name-Server, Option 119, Hostname
      Netbios-Name-Server, Netbios-Scope, MTU, Classless-Static-Route
      NTP

```

We now review the relay messages arriving on the DHCP server with the embedded information from the fabric relay agent such as the gateway IP and option 82 like the previously forwarded discover message:

```

10:14:30.778142 2c:6b:f5:f8:9b:f0 > 52:54:00:73:46:1c, ethertype IPv4 (0x0800), length 394: (tos
0x0, ttl 64, id 19872, offset 0, flags [none], proto UDP (17), length 380)
  10.99.99.3.67 > 10.91.91.42.67: BOOTP/DHCP, Request from 52:54:00:ae:1e:5b, length 352, xid
0xb7d35449, Flags [none]
    Gateway-IP 10.99.99.3
    Client-Ethernet-Address 52:54:00:ae:1e:5b
    Vendor-rfc1048 Extensions
      Magic Cookie 0x63825363
      DHCP-Message Option 53, length 1: Request
      Server-ID Option 54, length 4: 10.99.99.3
      Requested-IP Option 50, length 4: 10.99.99.63
      Hostname Option 12, length 8: "desktop1"
      Parameter-Request Option 55, length 13:
        Subnet-Mask, BR, Time-Zone, Default-Gateway
        Domain-Name, Domain-Name-Server, Option 119, Hostname
        Netbios-Name-Server, Netbios-Scope, MTU, Classless-Static-Route
        NTP
    Agent-Information Option 82, length 50:
      Circuit-ID SubOption 1, length 4: 1099
      Unknown SubOption 5, length 4:
        0x0000: 0a63 6303
      Unknown SubOption 11, length 4:
        0x0000: 0a63 6303

```

```
Unknown SubOption 9, length 30:
0x0000: 0000 0a4c 1904 1749 5242 2d69 7262 2e31
0x0010: 3039 393a 7674 6570 2e33 3237 3639
```

The DHCP server responds with the lease acknowledgement back towards the fabric relay agent:

```
10:14:30.778534 52:54:00:73:46:1c > 00:00:5e:e4:04:43, ethertype IPv4 (0x0800), length 384: (tos
0x0, ttl 128, id 47993, offset 0, flags [none], proto UDP (17), length 370)
10.91.91.42.67 > 10.99.99.3.67: BOOTP/DHCP, Reply, length 342, xid 0xb7d35449, Flags [none]
Your-IP 10.99.99.63
Gateway-IP 10.99.99.3
Client-Ethernet-Address 52:54:00:ae:1e:5b
Vendor-rfc1048 Extensions
Magic Cookie 0x63825363
DHCP-Message Option 53, length 1: ACK
RN Option 58, length 4: 345600
RB Option 59, length 4: 604800
Lease-Time Option 51, length 4: 691200
Server-ID Option 54, length 4: 10.99.99.3
Subnet-Mask Option 1, length 4: 255.255.255.0
Default-Gateway Option 3, length 4: 10.99.99.1
Domain-Name-Server Option 6, length 8: 8.8.8.8,9.9.9.9
Agent-Information Option 82, length 50:
Circuit-ID SubOption 1, length 4: 1099
Unknown SubOption 5, length 4:
0x0000: 0a63 6303
Unknown SubOption 11, length 4:
0x0000: 0a63 6303
Unknown SubOption 9, length 30:
0x0000: 0000 0a4c 1904 1749 5242 2d69 7262 2e31
0x0010: 3039 393a 7674 6570 2e33 3237 3639
```

The fabric relay agent filters some of the information out (such as option 82) and forwards the lease acknowledgement as Layer 2 unicast to the client:

```
10:14:30.780034 2c:6b:f5:f8:9b:f0 > 52:54:00:ae:1e:5b, ethertype IPv4 (0x0800), length 332: (tos
0x0, ttl 64, id 19875, offset 0, flags [none], proto UDP (17), length 318)
10.99.99.3.67 > 10.99.99.63.68: BOOTP/DHCP, Reply, length 290, xid 0xb7d35449, Flags [none]
Your-IP 10.99.99.63
Client-Ethernet-Address 52:54:00:ae:1e:5b
Vendor-rfc1048 Extensions
```

```

Magic Cookie 0x63825363
DHCP-Message Option 53, length 1: ACK
RN Option 58, length 4: 345600
RB Option 59, length 4: 604800
Lease-Time Option 51, length 4: 691200
Server-ID Option 54, length 4: 10.99.99.3
Subnet-Mask Option 1, length 4: 255.255.255.0
Default-Gateway Option 3, length 4: 10.99.99.1
Domain-Name-Server Option 6, length 8: 8.8.8.8,9.9.9.9

```

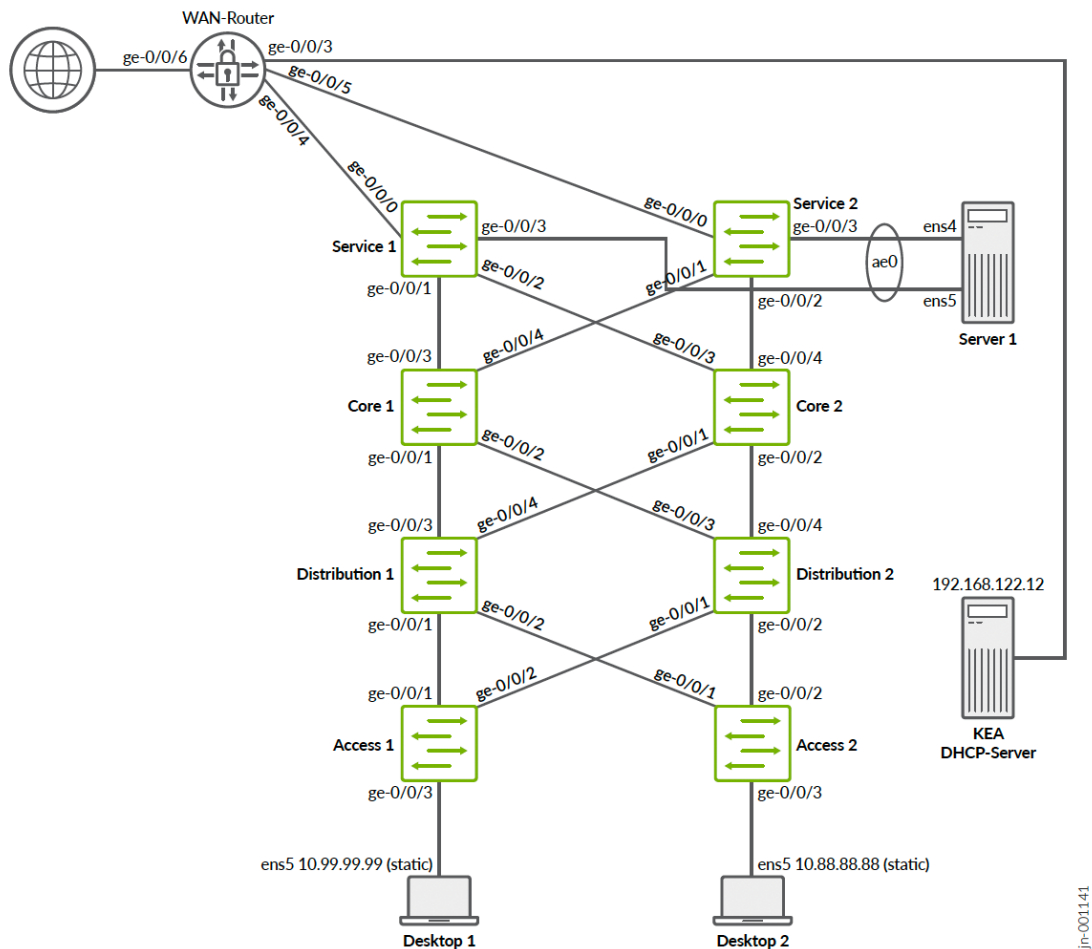
APPENDIX: Example DHCP Relay in IP Clos Fabric

IN THIS SECTION

- [Campus Fabric Dialogue Configuration | 51](#)
- [Access Switch Configuration | 54](#)
- [WAN Router Integration on the Service Block Switch | 56](#)
- [WAN Router Integration Checks | 57](#)
- [DHCP Server Check | 59](#)
- [Final Test with a Wired Client | 60](#)

Following is an example lab design to test DHCP relay in an IP Clos anycast fabric with the following configuration:

- Fabric type = IP Clos
- Overlay loopback pool configured = 172.16.192.0/19
- WAN router integration = eBGP peering with the fabric
- DHCP server location = External to the fabric
- DHCP server reachability = via WAN router for all VLANs and VRFs
- Third-party DHCP server used = Linux-based KEA V2.0.2 as VM



The configuration example shown below is only showing configuration relevant to the DHCP relay integration to make it brief. The full workflow can be deduced from available JVDs and extensions posted already.

Campus Fabric Dialogue Configuration


In the campus fabric dialogue configuration, it is important to configure the following:

- The correct fabric type = IP Clos
- The overlay loopback pool = 172.16.192.0/19 (as we expected some future growth)


Choose Campus Fabric Topology

Choose the topology you want to construct and configure related options


TOPOLOGY TYPE



EVPN Multihoming
Collapsed core with ESI-Lag



Campus Fabric Core-Distribution
EVPN core/distribution with ESI-Lag



Campus Fabric IP Clos
Campus fabric with L3 at the edge

CONFIGURATION

Topology Name

IP-Clos Fabric

TOPOLOGY SETTINGS

BGP Local AS

65001

(2-byte or 4-byte)

Underlay

☒ IPv4 ☐ IPv6

Subnet ⓘ

10.255.240.0/20

(xxx.xxx.xxx.xxx/xx)

Auto Router ID Subnet / Loopback Interface ⓘ

172.16.254.0/23

(xxx.xxx.xxx.xxx/xx)

Loopback Per-VRF Subnet ⓘ

172.16.192.0/19

(xxx.xxx.xxx.xxx/xx)

Then, in the “Networks” fabric dialogue, configure the following:

- DHCP Relay = Enabled
- vlan1033:
 - Network = vlan1033
 - DHCP Servers = 192.168.122.12
- vlan1088:
 - Network = vlan1088
 - DHCP Servers = 192.168.122.12
- vlan1099:

- Network = vlan1099
- DHCP Servers = 192.168.122.12

Configure Networks

Define networks, routing options, and port configurations

NETWORKS

VLANs for use in Campus Fabric topology

vlan1031	1031	>
vlan1033	1033	>
vlan1081	1081	>
vlan1088	1088	>

[Create New Network](#) [Add Existing Network](#)

OTHER IP CONFIGURATION

Network-specific IP configuration for each Access switch

access1	6 Static	>
access2	6 Static	>

VRF

Configuration

☒ Enabled ☐ Disabled

Instances

customera	2 networks	>
customerb	2 networks	>
devices	2 networks	>

[Add VRF Instance](#)

DHCP RELAY

☒ Enabled ☐ Disabled

DHCP Networks

vlan1033	relay	>
vlan1088	relay	>
vlan1099	relay	>

[Add DHCP Network](#)

Edit DHCP Network

Network

vlan1099 1099

DHCP Servers

192.168.122.12

(Comma-separated IPs(v4 or v6))

NOTE: Ensure you always use this dialogue to configure DHCP relay in all campus fabric designs.

Access Switch Configuration

Our Desktop1 client is attached to the ge-0/0/3 interface on the Access1 switch through the following port configuration:

PORT CONFIGURATION

Port Profile Assignment
★ Site, Template, or System Defined

Edit Port Configuration

Port IDs
ge-0/0/3
(ge-0/0/1, ge-0/0/4, ge-0/1/1-23, etc)

Interface
☒ L2 interface
 ☐ L3 interface
 ☐ L3 sub-interfaces

Configuration Profile
 vlan1099
 vlan1099(1099), access

☐ Enable Dynamic Port Configuration

Based on this configuration, Juniper Mist cloud configures the following on the switch:

```

set groups top routing-instances customera instance-type vrf
set groups top routing-instances customera interface irb.1091
set groups top routing-instances customera interface irb.1099
set groups top routing-instances customera forwarding-options dhcp-relay server-group vlan1099
192.168.122.12
    set groups top routing-instances customera forwarding-options dhcp-relay group
vlan1099 interface irb.1099
    set groups top routing-instances customera forwarding-options dhcp-relay group
vlan1099 active-server-group vlan1099
    set groups top routing-instances customera forwarding-options dhcp-relay group
vlan1099 relay-option-82 circuit-id vlan-id-only
    set groups top routing-instances customera forwarding-options dhcp-relay group
vlan1099 relay-option-82 server-id-override
    set groups top routing-instances customera forwarding-options dhcp-relay group
vlan1099 route-suppression destination
    set groups top routing-instances customera forwarding-options dhcp-relay group
vlan1099 overrides relay-source lo0.1
    set groups top routing-instances customera forwarding-options dhcp-relay forward-
only
set groups top routing-instances customera route-distinguisher 172.16.254.8:101
set groups top routing-instances customera vrf-target target:65000:101
  
```



```

set groups top routing-instances customera vrf-table-label
set groups top routing-instances customera routing-options auto-export
set groups top routing-instances customera routing-options multipath
set groups top routing-instances customera protocols evpn ip-prefix-routes advertise direct-
nexthop
set groups top routing-instances customera protocols evpn ip-prefix-routes encapsulation vxlan
set groups top routing-instances customera protocols evpn ip-prefix-routes vni 11299807
set groups top routing-instances customera protocols evpn ip-prefix-routes export
evpn_export_type5
    set groups top routing-instances customera interface lo0.1
.
set interfaces irb unit 1099 family inet address 10.99.99.1/24
set interfaces irb unit 1099 family inet mtu 9000
set interfaces irb unit 1099 description vlan1099
set interfaces irb unit 1099 no-dhcp-flood
set interfaces irb unit 1099 mac 00:00:5e:e4:31:57
.
set groups top interfaces lo0 unit 1 family inet address 172.16.192.1/32
.
set groups top policy-options policy-statement evpn_export_type5 term 01_ipv4 from protocol evpn
set groups top policy-options policy-statement evpn_export_type5 term 01_ipv4 from route-filter
0.0.0.0/0 prefix-length-range /32-/32
set groups top policy-options policy-statement evpn_export_type5 term 01_ipv4 then accept
set groups top policy-options policy-statement evpn_export_type5 term 02_direct from protocol
direct
set groups top policy-options policy-statement evpn_export_type5 term 02_direct then accept

```

The important item to verify on the access switch is which overlay loopback IP addresses have been assigned for each local VRF (Layer 3 is on the access switch in IP Clos):

STATISTICS	
STATUS	Connected
IP ADDRESS	<ul style="list-style-type: none"> 192.168.10.205 (fxp0.0) 172.16.254.8 (lo0.0) 172.16.192.1 (lo0.1) 172.16.192.2 (lo0.2) 172.16.192.3 (lo0.3) 10.33.31.1 (vlan 1031) 10.33.33.1 (vlan 1033) 10.88.81.1 (vlan 1081) 10.88.88.1 (vlan 1088) 10.99.91.1 (vlan 1091) 10.99.99.1 (vlan 1099) 10.255.240.25 (ge-0/0/1.0)
MIST APS	0

NOTE: You must ensure these local overlay loopback IP addresses are exchanged as host routes with the WAN router.

WAN Router Integration on the Service Block Switch

Now, check the BGP configuration on the two service block switches. Here, it is important to check the export filters (which are shared in our case):

BGP						
<input checked="" type="radio"/> Enabled <input type="radio"/> Disabled						
<input type="text" value="Search"/> Add BGP Group						
3 BGP Groups						
NAME	TYPE	LOCAL AS	EXPORT	IMPORT	NEIGHBORS	NEIGHBORS AS
customer0	external	64911	export-vrfs	import-default	1	64901
customerb0	external	64911	export-vrfs	import-default	1	64901
devices0	external	64911	export-vrfs	import-default	1	64901

It is important to also export the overlay loopback IP address range as well as the usual VLANs of the fabric.

Edit Routing Policy			
Name			
export-vrfs			
TERMS Add Terms			
Name	Prefix	AS Path	Protocol
vlan1091	10.99.91.0/24	--	--
vlan1099	10.99.99.0/24	--	--
vlan1081	10.88.81.0/24	--	--
vlan1088	10.88.88.0/24	--	--
vlan1031	10.33.31.0/24	--	--
vlan1033	10.33.33.0/24	--	--
overlaylo0	172.16.192.0/19-32	--	--

NOTE: You must ensure that you append “-32” to the IP prefix that you have defined in the campus fabric dialogue. Otherwise, the WAN router does not know the individual host routes which are spread across your VRFs.

WAN Router Integration Checks

Next, login to the WAN router and verify the received overlay loopback IP addresses. First, check the established BGP peering:

```
root@wanrouter> show bgp summary
Threading mode: BGP I/O
Default eBGP mode: advertise - accept, receive - accept
Groups: 3 Peers: 6 Down peers: 0
Peer          AS      InPkt   OutPkt   OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
10.255.224.1   64911   5685    5650     0      0 1d 19:02:18 Establ
  public-int.inet.0: 5/6/6/0
10.255.224.3   64911   5680    5649     0      0 1d 19:02:12 Establ
  public-int.inet.0: 5/6/6/0
10.255.224.5   64911   5685    5649     0      0 1d 19:02:16 Establ
  public-int.inet.0: 5/6/6/0
10.255.226.1   64911   5686    5648     0      0 1d 19:02:14 Establ
  public-int.inet.0: 5/6/6/0
10.255.226.3   64911   5683    5649     0      0 1d 19:02:19 Establ
  public-int.inet.0: 5/6/6/0
10.255.226.5   64911   5686    5649     0      0 1d 19:02:22 Establ
  public-int.inet.0: 5/6/6/0
```

Then, check the routing table for the three IP addresses (172.16.192.1-3) that we validated on the local switch shown above:

```
root@wanrouter> show route table public-int.inet.0
public-int.inet.0: 35 destinations, 53 routes (35 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
0.0.0.0/0      *[Static/5] 1d 18:59:15
                > to 192.168.10.1 via ge-0/0/6.0
```

```

.
10.99.99.0/24    *[BGP/170] 00:50:33, localpref 100
                 AS path: 64911 65002 65004 65006 65007 I, validation-state: unverified
                 to 10.255.224.1 via ge-0/0/4.1091
                 > to 10.255.226.1 via ge-0/0/5.1091
                 [BGP/170] 00:50:32, localpref 100
                 AS path: 64911 65001 65004 65006 65007 I, validation-state: unverified
                 > to 10.255.224.1 via ge-0/0/4.1091
.
172.16.192.1/32 *[BGP/170] 00:50:36, localpref 100
                 AS path: 64911 65002 65004 65005 65008 I, validation-state: unverified
                 to 10.255.224.1 via ge-0/0/4.1091
                 > to 10.255.226.1 via ge-0/0/5.1091
                 [BGP/170] 00:50:36, localpref 100
                 AS path: 64911 65001 65004 65005 65008 I, validation-state: unverified
                 > to 10.255.224.1 via ge-0/0/4.1091
172.16.192.2/32 *[BGP/170] 00:50:36, localpref 100
                 AS path: 64911 65002 65004 65005 65008 I, validation-state: unverified
                 to 10.255.224.3 via ge-0/0/4.1081
                 > to 10.255.226.3 via ge-0/0/5.1081
                 [BGP/170] 00:50:36, localpref 100
                 AS path: 64911 65001 65004 65005 65008 I, validation-state: unverified
                 > to 10.255.224.3 via ge-0/0/4.1081
172.16.192.3/32 *[BGP/170] 00:50:36, localpref 100
                 AS path: 64911 65002 65004 65005 65008 I, validation-state: unverified
                 to 10.255.224.5 via ge-0/0/4.1031
                 > to 10.255.226.5 via ge-0/0/5.1031
                 [BGP/170] 00:50:36, localpref 100
                 AS path: 64911 65001 65004 65005 65008 I, validation-state: unverified
                 > to 10.255.224.5 via ge-0/0/4.1031
.
192.168.10.0/24  *[Direct/0] 1d 18:59:15
                 > via ge-0/0/6.0
192.168.10.59/32 *[Local/0] 1d 18:59:15
                 Local via ge-0/0/6.0
192.168.122.0/24 *[Direct/0] 1d 18:59:15
                 > via ge-0/0/3.0
192.168.122.23/32 *[Local/0] 1d 18:59:15
                 Local via ge-0/0/3.0

```

DHCP Server Check

Next, check the DHCP server itself:

```
root@kea1:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen
1000
    link/ether 52:54:00:18:ee:80 brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    inet 192.168.122.12/24 brd 192.168.122.255 scope global ens3
        valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe18:ee80/64 scope link
        valid_lft forever preferred_lft forever
root@kea1:~# ip r
default via 192.168.122.1 dev ens3 proto static
10.0.0.0/8 via 192.168.122.23 dev ens3 proto static
172.16.192.0/19 via 192.168.122.23 dev ens3 proto static
192.168.122.0/24 dev ens3 proto kernel scope link src 192.168.122.12
root@kea1:~# netstat -tunap | grep 67
udp        0      0 192.168.122.12:67      0.0.0.0:*                747/kea-dhcp4
root@kea1:~# systemctl status kea-dhcp4-server.service
* kea-dhcp4-server.service - Kea IPv4 DHCP daemon
    Loaded: loaded (/lib/systemd/system/kea-dhcp4-server.service; enabled; ven>
    Active: active (running) since Mon 2024-06-24 15:52:41 UTC; 1 day 19h ago
    Docs: man:kea-dhcp4(8)
    Main PID: 747 (kea-dhcp4)
    Tasks: 5 (limit: 1012)
    Memory: 13.6M
    CPU: 8.078s
    CGroup: /system.slice/kea-dhcp4-server.service
            └─747 /usr/sbin/kea-dhcp4 -c /etc/kea/kea-dhcp4.conf
```

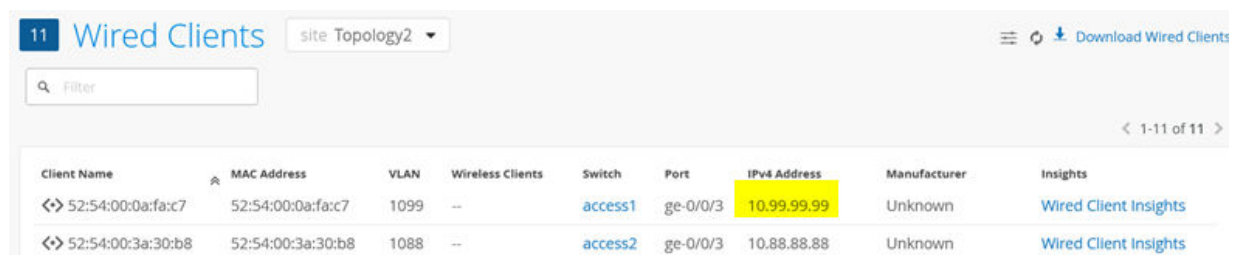
Final Test with a Wired Client

To complete the installation, perform a final test with a wired client. The initial state of our client is that it has a static IP address assigned and can communicate towards the internet:

```
root@lab2-desktop1:~# ifconfig ens5
ens5: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.99.99.99 netmask 255.255.255.0 broadcast 10.99.99.255
    inet6 fe80::5054:ff:fe0a:fac7 prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:0a:fa:c7 txqueuelen 1000 (Ethernet)
    RX packets 23 bytes 11662 (11.6 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 65 bytes 7564 (7.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@lab2-desktop1:~# ping -c3 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=55 time=26.0 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=55 time=36.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=55 time=39.6 ms
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 26.044/34.028/39.584/5.788 ms
```

We can see this client and its IP address in the Wired Client overview in the Juniper Mist portal:



Client Name	MAC Address	VLAN	Wireless Clients	Switch	Port	IPv4 Address	Manufacturer	Insights
<> 52:54:00:0a:fa:c7	52:54:00:0a:fa:c7	1099	--	access1	ge-0/0/3	10.99.99.99	Unknown	Wired Client Insights
<> 52:54:00:3a:30:b8	52:54:00:3a:30:b8	1088	--	access2	ge-0/0/3	10.88.88.88	Unknown	Wired Client Insights

Next, we unconfigure the static IP address and try to obtain a DHCP lease instead. The additional configuration below ensures that the client loses the static configuration and any prior knowledge of a DHCP lease. We then start up the DHCP client in the foreground to see a bit more of the debugging messages:

```
root@lab2-desktop1:~# ip link set ens5 up
root@lab2-desktop1:~# ip addr del 10.99.99.99 dev ens5
root@lab2-desktop1:~# pkill dhclient
root@lab2-desktop1:~# rm -f /var/lib/dhcp/*.leases
```

```

root@lab2-desktop1:~# dhclient -v ens5
Internet Systems Consortium DHCP Client 4.4.1
Copyright 2004-2018 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
Listening on LPF/ens5/52:54:00:0a:fa:c7
Sending on   LPF/ens5/52:54:00:0a:fa:c7
Sending on   Socket/fallback
DHCPDISCOVER on ens5 to 255.255.255.255 port 67 interval 3 (xid=0x9778d1e)
DHCPOFFER of 10.99.99.10 from 172.16.192.1
DHCPREQUEST for 10.99.99.10 on ens5 to 255.255.255.255 port 67 (xid=0x1e8d7709)
DHCPACK of 10.99.99.10 from 172.16.192.1 (xid=0x9778d1e)
bound to 10.99.99.10 -- renewal in 747 seconds.

```

After a while (depending on local ARP age-outs), this change becomes visible in the Wired Client overview:

Client Name	MAC Address	VLAN	Wireless Clients	Switch	Port	IPv4 Address	Manufacturer	Insights
<> 52:54:00:0a:fa:c7	52:54:00:0a:fa:c7	1099	--	access1	ge-0/0/3	10.99.99.10	Unknown	Wired Client Insights
<> 52:54:00:3a:30:b8	52:54:00:3a:30:b8	1088	--	access2	ge-0/0/3	10.88.88.88	Unknown	Wired Client Insights

APPENDIX: Fabrics with Overlay Loopback Usage Before June 20, 2024, Deployed

NOTE: This chapter is only valid for campus fabrics deployed before June 20, 2024, and that use the overlay loopback IP address approach.

When using a campus fabric deployed before June 20, 2024, the configuration to propagate the loopback address host routes is not automatically added by Juniper Mist cloud. In this case, you can simply use Additional CLI Commands as shown in the example. In the case of an IP Clos fabric where the

VRFs are located on the access switch, use the following configuration statements **on every access switch**, adapting the names of the example VRFs.

```
set groups top policy-options prefix-list irb_networks apply-path "interfaces irb unit <*>"
family inet address <*>"
set groups top policy-options prefix-list loopbacks apply-path "interfaces lo0 unit <*>" family
inet address <*>"
set groups top policy-options policy-statement EXPORT-T5-ROUTES term TERM_LO from protocol direct
set groups top policy-options policy-statement EXPORT-T5-ROUTES term TERM_LO from prefix-list
loopbacks
set groups top policy-options policy-statement EXPORT-T5-ROUTES term TERM_LO then accept
set groups top policy-options policy-statement EXPORT-T5-ROUTES term TERM_IRB from protocol
direct
set groups top policy-options policy-statement EXPORT-T5-ROUTES term TERM_IRB from prefix-list
irb_networks
set groups top policy-options policy-statement EXPORT-T5-ROUTES term TERM_IRB then accept
set groups top routing-instances customera protocols evpn ip-prefix-routes export EXPORT-T5-
ROUTES
set groups top routing-instances customerb protocols evpn ip-prefix-routes export EXPORT-T5-
ROUTES
set groups top routing-instances devices protocols evpn ip-prefix-routes export EXPORT-T5-ROUTES
```

On the service block functions where you make the integration of the WAN router, you need to repeat the above and depending on the protocol towards the WAN router also **add the following** configuration:

```
set groups top policy-options policy-statement EXPORT-T5-ROUTES term TERM_BGP from protocol bgp
set groups top policy-options policy-statement EXPORT-T5-ROUTES term TERM_BGP then accept
```

When using OSPF, use the following statements instead:

```
set groups top policy-options policy-statement EXPORT-T5-ROUTES term TERM_OSPF from protocol ospf
set groups top policy-options policy-statement EXPORT-T5-ROUTES term TERM_OSPF then accept
```

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners. Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice. Copyright © 2025 Juniper Networks, Inc. All rights reserved.