Juniper Networks | Engineering Simplicity

# Campus Fabric WAN Router Integration—Juniper Validated Design Extension (JVDE)

Published
2025-08-20

# Table of Contents

# Campus Fabric WAN Router Integration—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.

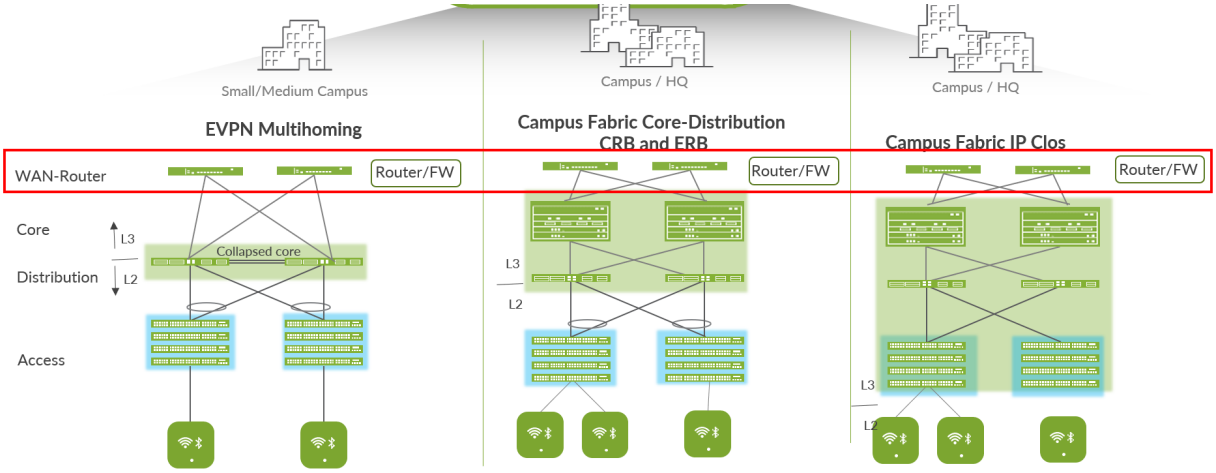
# About this Document

When building any EVPN Fabric, it is necessary to integrate the WAN routers. This does not change even when using any Mist managed Campus Fabric which provides a fast and easy way to build the fabric itself. After building the campus fabric you must perform the mandatory step of WAN router integration.

By design the WAN router itself can be an MX-Router or an SRX-Series Firewall from Juniper but it can also be a Router/Firewall from a Third-Party Vendor. This JVDE describes the various ways of WAN router integration and the testcases that were performed to ensure proper integration. We provide a list of necessary needs for features from Third-Party WAN router vendors that are mandated for integration. Furthermore, complete configuration examples using the Mist GUI (and configurations on a Juniper MX-Router) are provided in the Appendix Section for reference.

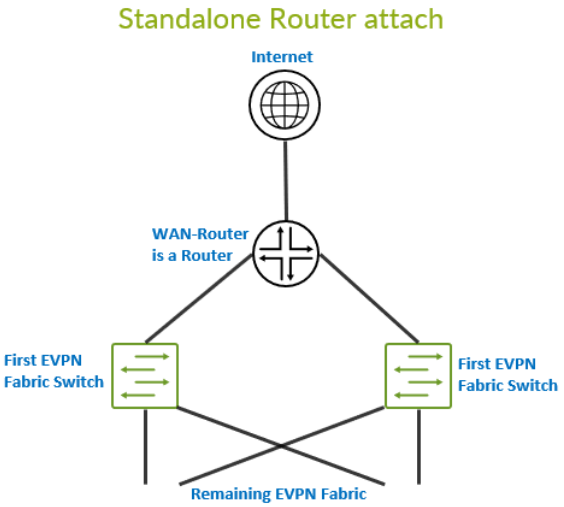
# Solution Benefits and Overview

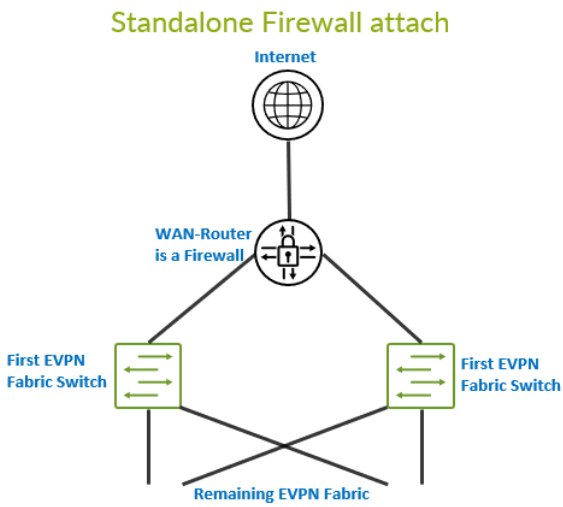
When building any Juniper Campus Fabric, you must always plan for the integration of a WAN router since we do not assume deployment in an air-gapped environment. You must be able to have the fabric connected to the Internet. Therefore, WAN router integration is a mandatory part of the fabric installation procedure.




There are multiple ways to integrate a WAN router depending on the function it performs and if it just should be a standalone or a redundant design for a classical router or a stateful firewall.

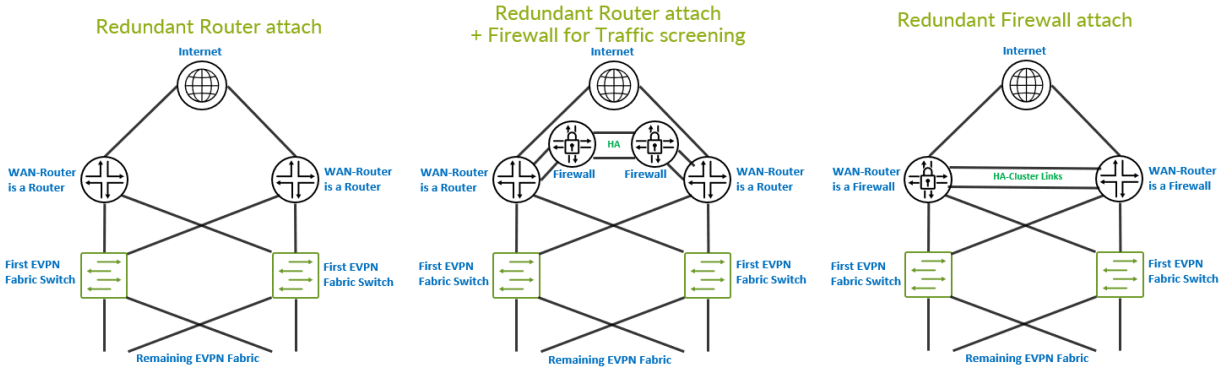
For a lab design, you can choose to use a Juniper MX Series router or SRX Series Firewall or any third-party router or firewall. Just attach the device to the first two switches of the fabric such as the service block switch, core switch or collapsed core switch.




A more robust design intended for production grade fabrics is a to use a redundant pair of WAN routers as shown. A third-party firewall vendor must provide stateful redundancy synchronization because the fabric may be designed to use other links to forward traffic. The vendor-specific implementation is usually a pair of redundant links between the firewall devices.

**Figure 1: Redundant WAN Router Fabric Design**




It is always highly recommended to review the implementation of the WAN router as part of the Campus Fabric Rollout planning. There needs to be a creation level of feature support to make this a successful design and there are different flavours to choose. This JVDE reviews the integration options and guides customer to select the appropriate method for integration.

# Use Case and Reference Architecture

**IN THIS SECTION**



Before we go deeper into the integration, we must review how the fabric usually operates and forwards Packets. It is critical to understand how a Layer 2 (L2) VLAN at an Access-Switch of the fabric, where Wired and Wireless Client are connected to, makes its way to other VLANs with Clients and forward and also via the WAN router towards internet.

## Basic Forwarding Operation of a Fabric

Without corner cases like the seldom used Bridged-Overlay-option it can be assumed that all VLANs of a fabric are connected to at least one global Virtual Routing Function (VRF). Most customers use a handful of multiple VRFs. By design:

- The fabric VRF will contain the IP address as Gateway for each attached VLAN Client at Access Switch. The Wired Client needs to know and send Traffic to that IP address if he wants to leave the fabric or communicate with any other VLAN in the fabric. This can be a static assignment on the Client side, or he obtains this information via a DHCP-Lease.
- VLAN's that are connected to the same VRF can talk directly to each other and exchange Traffic. This type of East-West Traffic is always handled directly in the fabric. If the Traffic needs to be controlled inside a VRF then typically ACLs or, if the fabric supports it, VXLAN Group Based Policies can be used deployed on the Access Switch.
- VRF's are ALWAYS isolated! This is a security measurement by Design. As a result, all Traffic between VRF's need to go up to the WAN router for security screening. Should the WAN router then permit this Traffic then it will be sent back to the next VRF to the fabric. So, one always enforces North-South Traffic this way.

## Virtual Gateway Fabric Versus Anycast Fabric

Depending on the fabric Type the Overlay VLAN's, where the Client Traffic is in, may need additional IP addresses for internal proposals, which is the case for Virtual Gateway Fabrics. Mist Campus Fabrics configures the following fabric Types:

| Fabric Type | Virtual Gateway Fabric | Anycast Fabric |
|---|---|---|
| EVPN Multihoming Fabric | Yes | --- |
| Central Routed and Bridged Fabric (CRB) | Yes | --- |
| Edge Routed and Bridged Fabric (ERB) | --- | Yes |
| IP-Clos Fabric | --- | Yes |

In a Virtual Gateway Fabric, you typically have a very limited amount of VRF's located in the fabric. Those are located at the Core or Collapsed Core Switches. As the maximum supported amount of Core/ Collapsed Core Switches in a Mist Campus Fabric can manage is four. This also means a certain VRF can

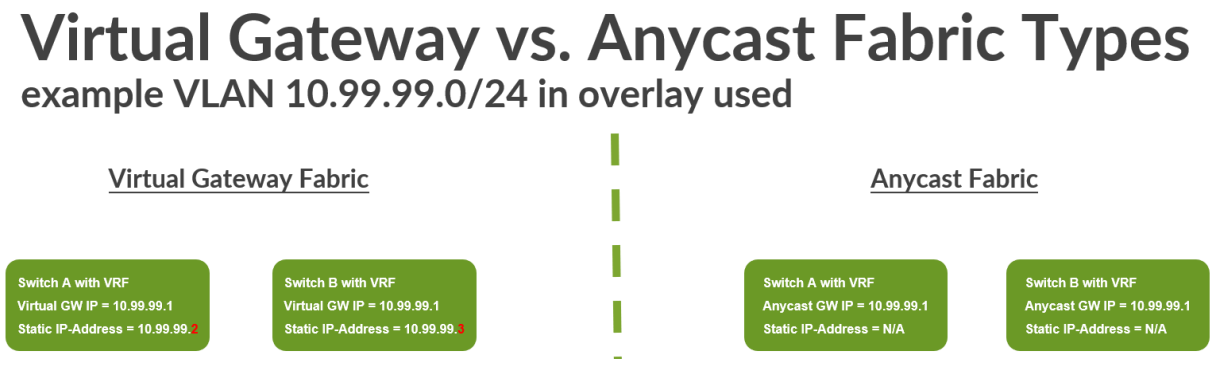
be duplicated to each redundant Core/Collapsed-Core Switch maximum four times in the fabric. Anycast Fabrics, however, are designed for more scale out designs hence the location of the VRFs is either on the Distribution Switch (ERB) or in fact even at the Access Switch (IP-Clos). The nature of all Virtual Gateway Fabrics is however that the system also assigns for every VLAN located in the fabric an additional static IP address that is unique per VRF. Hence apart from the Gateway IP address of the gateway for a VLAN you need in each subnet up to four additional IP addresses as the maximum number Core/Collapsed Core Switches in a Mist Campus Fabric is 4.

Why such a design? Well, there are benefits for certain Traffic of the fabric, for example, when doing DHCP-Relay. For DHCP-Relay the system uses the static IP address instead of the Gateway IP address when forwarding the DHCP-Client requests from a certain VRF. This behavior will ensure that the returned answer will be send back straight to the concerned VRF as the static IP address is unique to the VLAN/Core-Switch (with VRF).

Another way to think about a Virtual Gateway Fabric is if you compare it with traditional L2 Gateway failover designs such as VRRP. There you always have a VIP which floats between the Gateways (that are our VRF's) and each Gateway always needs an additional unique static IP only he has in that VLAN. In a Mist Campus Fabric there is naturally no VRRP protocol ever spoken or needed as the EVPN control plane takes over for it.

This small sacrifice needing to plan room for those additional static IP addresses in each VLAN is eliminated in Anycast Fabrics. This is because the more scale out 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.

**NOTE**: When creating a new VLAN, the fabric gateway IP address is the lowest host IP address of a subnet. In case of a virtual gateway fabric, the maximum 4 additional static IP addresses needed are usually increments of that lowest host IP address of a subnet. As a best practice doing manual changes to the gateway IP addresses and static IP addresses should be avoided. Changing those addresses only adds confusion when others might also manage the fabric.



## Service Block Function of a Fabric

When designing the connection of the fabric to the WAN router, you will leverage a so-called service block function. You may also find the terms service leaf or border leaf used in other literature. The service block function is meant for all kind of integration scenarios such as:

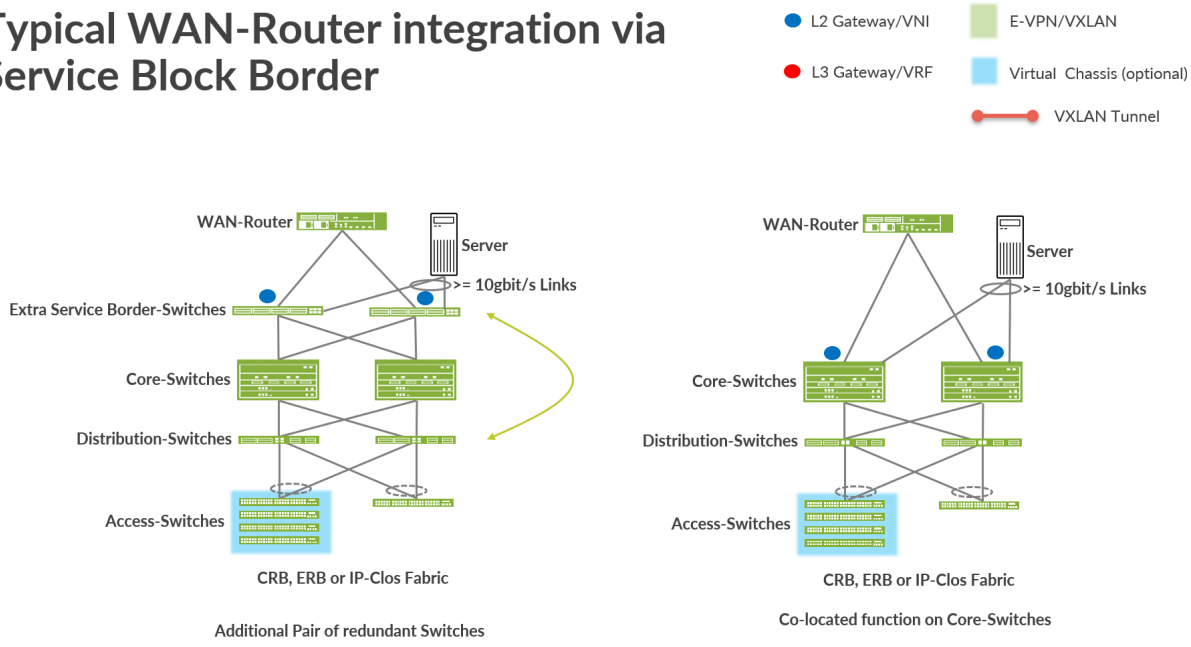
- WAN router integration in a fabric.
- Server attachment for any kind of local services your fabric needs to provide such as:
  - DHCP server for fabric VLANs.
  - File services
  - Webservers
  - Mist edge for wireless network overlay
  - Many more services
- All kinds of migration scenarios with legacy fabrics and network designs

A service block function in a Mist Fabric can be either virtual or physical depending on the design of the fabric.

- A virtual service block function is a co-located function that is usually added to a fabric node that usually has a different function. In our case that means that the Service block function is added to the Core-Switches of the fabric. This is the default design that enables you to deploy a fabric with minimal Hardware footprint.
- A physical Service block function is always consistent of a pair of dedicated Switches on-top of the Core-Switches of a fabric deployed. You can think of those as a pair of dedicated Distribution Switches swapped northbound of the fabric. Hence it is also recommended to use similar Hardware as your distribution Switches have.

The picture below shows both designs where the physical service block function is on the left and the virtual service block function is on the right side of the picture.

## Typical WAN-Router integration via Service Block Border




When deploying fabric, it's typically the scale, local port or interface usage, and port speed and density of the core switches that influence the decision between using a virtual or a physical switch deployment. We recommend that you consider the following:

- Apart from the WAN routers do you have enough ports left for future server attachment?
- Do the supported port speeds match with what you want to attach?
- Can a physical service block function present a scale limit in the future?
- The speed of the connected WAN router. Remember that all VRF-to-VRF traffic must go through the WAN router by design.

**NOTE**: When the fabric grows in the future to more than two core switches use then you must have a dedicated pair of physical service block switches for the service block function.

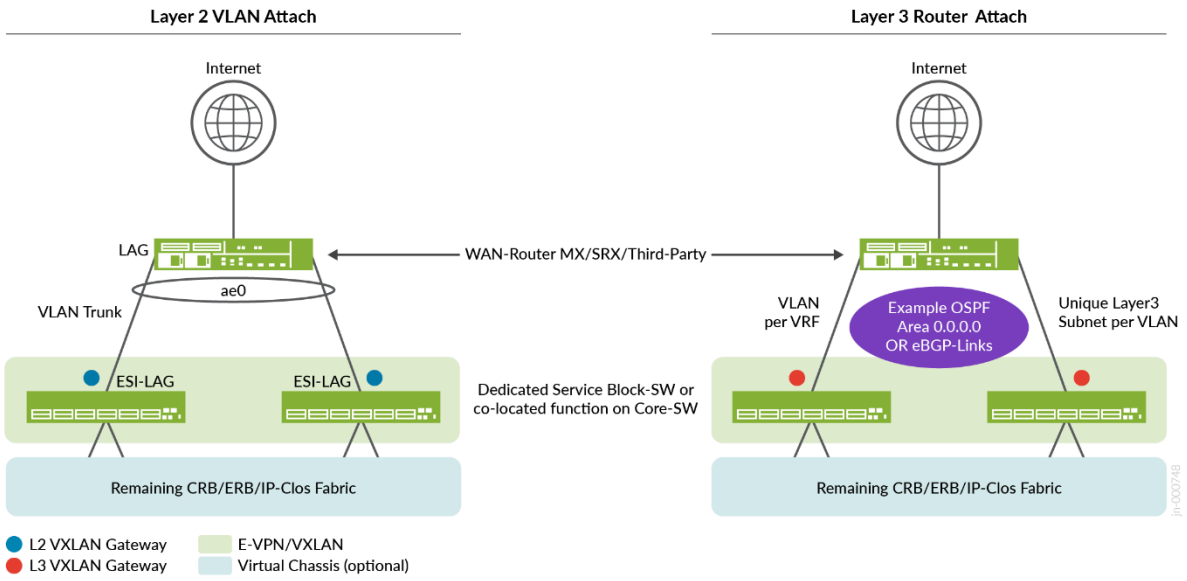
## WAN Router Integration Using Service Block Function

There are several ways one can attach a fabric to the WAN router for traffic towards the internet. Usually, one determines them by the way you attach the WAN router to the service block function of the fabric.

You can attach the WAN router:

- Using an L2 Method:
  - VLAN's seen on the access layer of a VRF or additional defined transport VLAN's attached to VRF will be shared via a Trunk-Interface towards WAN router.
  - The trunk links between the service block function of the fabric and WAN router must use IEEE 803.2ad LAG/LACP to be able to detect link-failures and missing devices. You can NOT use STP here. On the service block function of the fabric, you will configure an ESI-LAG to achieve this. The WAN router side just needs a standard IEEE 803.2ad LAG/LACP configuration with active Link management. Should the latter not be supported by the vendor of the WAN router then consider a Layer 3-Method instead.
  - In the Fabric dialogue for each VRF one must configure a manual Route with a static IP address for all default Traffic towards the Internet and other VRF's. This static IP address is then reachable via the WAN router.
  - WAN router needs to have this static IP address assigned to an interface towards service block function of the fabric for reachability. Redundancy of this static IP address needs to be achieved by means of an L2 Gateway redundancy Protocol. The usage of VRRP is highly recommended here and the static IP address is then the VIP.
  - On WAN router you may have to define an additional static Route for the VLAN's attached to a VRF.
- Using an L3 Method:
  - The links between the service block function of the fabric and WAN router will be defined as L3 P2P-Links with IP addresses. Those will need to be individually configured on each service block function of the fabric and opposite WAN router.
  - Each P2P Link will need to be assigned to a VLAN-Name that is ALSO assigned to the VRF of the fabric. Via this indirect linking the Mist Cloud can reference and bind a particular VRF to the Link. Additionally, the VLAN-ID assigned to the P2P Link provides isolation against other VRF's on the same Link.
  - In the Fabric dialogue for each VRF you do NOT need to configure any additional Route towards WAN router as the fabric gets this information from the WAN router via a supported Routing protocol.
  - You have to set up and create policies for import and exporting routes between fabric and WAN router.
  - You must use a L3 Routing Protocol to establish the route-exchange and forwarding between the fabric and WAN router. Supported as of today are:
    - Exterior BGP, which is fully UI, driven.
    - OSPF (Policies for im/export of Routes are today still needing additional CLI).

**NOTE**: RECOMMENDED is using the **L3 eBGP** based method wherever this is technically possible from the first day on. Even if it is initially a higher effort to configure, it will be the only method in the future to achieve new features such as DCI.




## L2 WAN Router Attach Details

**NOTE**: L2 WAN router attachment methods should only be chosen in Lab-designs or small fabrics. Even then if you design an HA/Redundant WAN router design the WAN router vendor MUST support IEEE 803.2ad LAG/LACP AND a redundant L2 Gateway Method such as VRRP. Without the two supported you won't be able to achieve a HA/Redundant WAN router design that does not fail at some point.

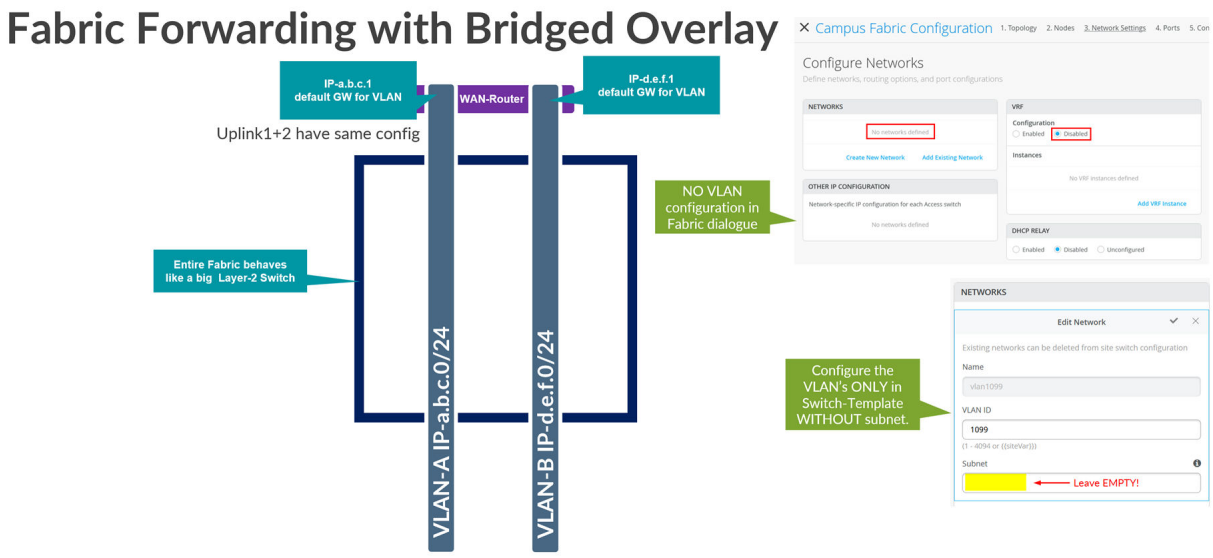
If you use an **L2** method**,** you have the following options:

- Treat the entire fabric as a big L2 Switch and use the Bridged Overlay model.
- Strech at least one VLAN from Access Switch side also to the WAN router and that the WAN router an IP address in that VLAN is then the Default-GW for the VRF itself. (Do not use for production.)
- Define a dedicated Transport VLAN for each VRF. (Recommended when doing L2 exit.)

The bridged overlay model allows you to handle all the Layer 3 (L3) gateway functions of a VLAN directly on the WAN router itself. Many technical drawbacks might prevent you from using certain features of the fabric. If you want to migrate directly from a legacy design, you can use the bridged overlay model, so we describe the uses and benefits of that model here:

- There are no VRF configured anywhere in the fabric. Therefore, the fabric acts like a large, distributed L2 switch.
- All traffic is anchored outside of the fabric at the external WAN router.
- WAN router must play the role of the GW in each VLAN.
- VLANs can only talk to each other VIA the WAN router. Hence any East-West Traffic is always enforced to go up to the WAN router first for any VLAN to VLAN Traffic.
- All VLANs you are using on any Access Port of the fabric have to be also configured on the Uplink Ports towards WAN router.

**Figure 2: Fabric Forwarding with Bridged Overlay**




The following lists the known limitations of the bridged overlay approach:

- There is a limit of around 250 VLANs you can use with this approach. This is mainly because the WAN router cannot provide more than 250 VRRP groups for gateway failover. You can confirm this with the WAN router vendor.
- If DHCP relay is required, then it must be configured on the WAN router.

- This model allows the WAN router to be the DHCP server for your VLANs. However, it also means that you must configure DHCP lease redundancy between two WAN routers when those WAN routers are deployed as an HA pair.
- All East-West inter-VLAN traffic on the fabric must flow through the WAN router.
- When implementing bridge overlay in an IP Clos fabric, some features may be lost. Since inter-VLAN traffic is always forced to exit the fabric—and VXLAN headers are stripped before reaching the WAN router—Group-Based Policies (GBP) become limited in functionality. As a result, GBP can only control or block traffic between clients within the same VLAN. It is not possible to manage traffic between clients in different VLANs, even if they belong to the same VRF.

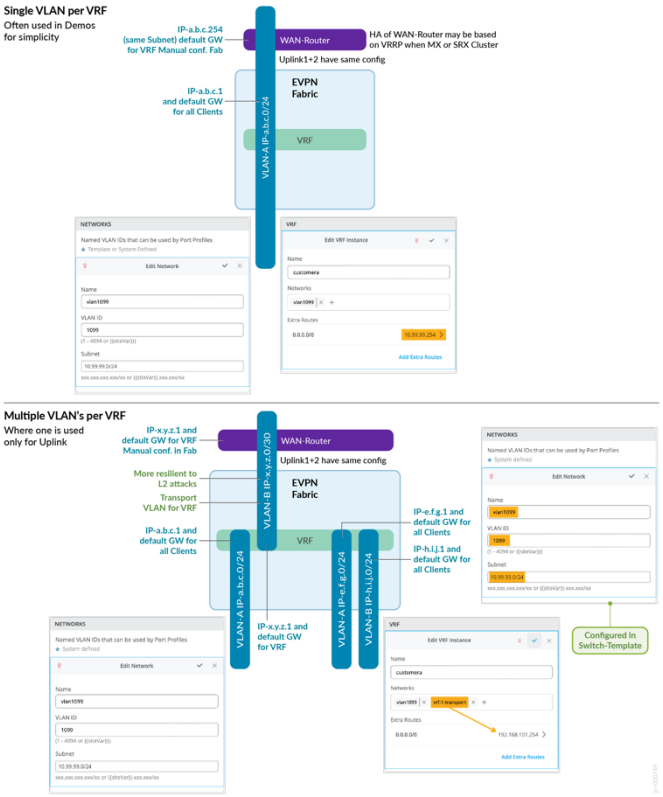
> **NOTE**: A combination of the bridged overlay method and some other method to achieve a hybrid design is technically possible. Such a combination is often used when customers decide to provide a separate path (VLAN) for guest access that must not interfere with the fabric VLANs used for regular clients.

Let's now review the other two most commonly used L2 models:

- **Stretched VLAN**: This is the method used in labs to make fast progress when attaching WAN routers; where the goal is not to provide a production-grade design but rather something simple and easy to debug. You see this method commonly used in Juniper JVDs and NCEs as an example.
  - Using this method, a VLAN within a VRF instance that is used in access switches will also be used on the uplink to the WAN router, between the service block function and the WAN router.
  - As a result of this stretched VLAN, the WAN router must be assigned a free IP address on that VLAN and the manual route for the default GW in the VRF configuration will point to that IP address.
  - You can attach more VLANs to that VRF if needed. But you must delete or modify the VLAN used for the stretch to the WAN router.
  - Do not use the stretched VLAN method in production environments. It has some downsides such as when a packet might need to be hair-pinned inside the fabric because of sub-optimal routing within the fabric.
- **Transport VLAN**: This is the method recommended for use in a production-grade design when using an L2 attachment method.
  - Using this method, a dedicated VLAN per VRF or WAN router must be used on the uplink to the WAN router between the service block function and the WAN router. This dedicated VLAN is not used on any access switch within the fabric.

- The WAN router is assigned a free IP address on that dedicated VLAN and the manual route for the default GW in the VRF configuration points to that IP address.
- In this case, it is assumed that you have one or more other VLANs that you use on the access switches for that VRF.

**Figure 3: Fabric Forwarding Using L2**




Going deeper into the forwarding and configuration of a stretched VLAN:

- On the service block border function, you create an ESI-LAG that contains all stretched VLANs that belong to the fabric (one per VRF).
- The WAN router only needs ordinary LAG support because with the stretched VLAN, you pool the links towards the attached service block border switches in a single LAG configuration.
- For example, on the VLAN 10.99.99.0/24, the fabric VRF might have the anycast or virtual GW IP address 10.99.99.1. This address will be used by all wired and wireless clients to send traffic to the fabric.
- In the VRF we configure a default route (0.0.0.0/0) with a gateway of 10.99.99.254

- The IP address 10.99.99.254 is then configured at vlan 1099 on the WAN router providing the forwarding for the fabric.
- If you have redundant WAN routers, then configure a VRRP-VIP with the address 10.99.99.254 on them.

Going deeper into the forwarding and configuration of a transport VLAN:

- Define additional transport VLANs in your switch template. Just define a VLAN-ID with no network information.
- In your fabric definition, exclude the transport VLANs in the VRF they should use. Only add the access VLANs there.
- Do not define a default route in the fabric VRFs. The default route gets configured in the service block definition.
- On the service block border switch, add the local IP address of the network for the VLAN, such as 192.168.101.1/24, using the additional IP address configuration.
- On the Service block border switch, create an ESI-LAG that contains only the chosen transport VLAN of each VRF.
- The WAN router only needs ordinary LAG support because you pool the links to the attached service block border switches in a single LAG configuration.
- You then need to manually create and edit the VRF service block border switch
  - Add your transport VLAN to the access-VLANs that automatically appear.
  - Create a default a route (0.0.0.0/0) with a gateway of 192.168.101.1.254
- For a VLAN such as 10.99.99.0/24 the access or fabric VRF will have the Anycast/virtual GW IP address which is in this case the 10.99.99.1. This will be used by Wired/Wireless Clients.
- The IP address 192.168.101.254 then is configured at vlan 101 on the WAN router providing the forwarding of the fabric.
- The WAN router must also configure a static route towards 10.99.99.0/24 via 192.168.101.1 as that is the Link to the VRF of the fabric.
- Should you have redundant WAN routers then configure VRRP-VIP for 192.168.101.254 on them.
- Finally, it is also recommended, using additional CLI at this point in time, that you change the transport VLANs and VRF's on the service block function into Virtual Gateway addressing independent of the fabric type! This will warrant that the Traffic will flow in the optimal Way. Should you forget to make this improvement there may be situations happening where Traffic unnecessary

hairpins twice your service block functions via the below distribution Switch. The situation happens in the following example when:

- A packet leaves the fabric out on service block function 1 via uplink1 as anchor-point for the ESI-LAG.
- A WAN router gets the packet on uplink1 but sends the answer packet down to uplink2 towards service block function 2.
- Service block function 2 sees that the anchor point was service block function 1 but has no direct link to it. So, it sends the answer packet down to a distribution switch.
- The distribution switch forwards the packet back up to the service block function 1.
- The service block function 1 then as anchor point forwards the answer to one of the distribution switches as it should be under normal conditions.

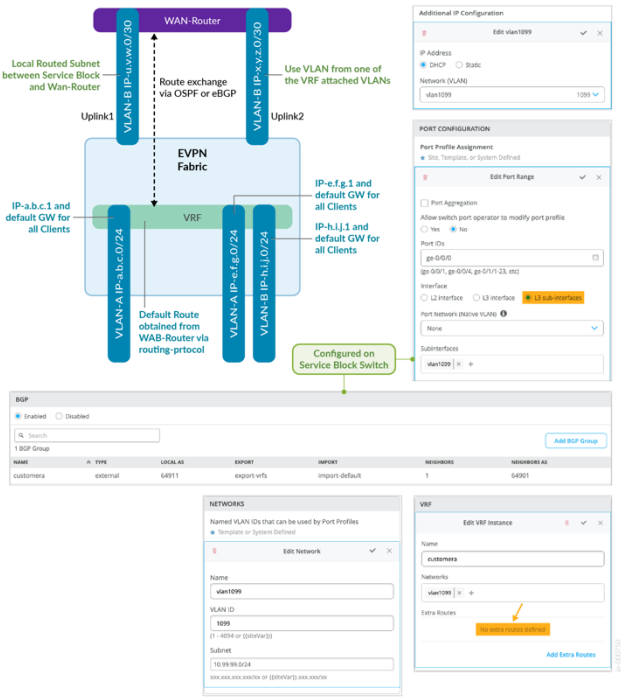
When we make the suggested change, this additional hair-pinning of traffic is avoided. Please see more how this is achieved in the appendix section below where we share concrete examples.

> **NOTE**: A common error is that people forget to sync the AE-Index-Field when defining the uplinks on the two service block functions of the fabric. The same values must be chosen on both service block function interfaces. The system does not check this and does not warn you if the fabric uses it elsewhere. Also, make sure to enable the **esilag** configuration knob.

## L3 WAN Router Attach Details

Finally, let's review the more robust and scalable L3 methods of attaching a WAN router to a fabric. When you use an **L3** method**,** you have the following options:

- Use OSPF as the routing protocol between the fabric and the WAN router.
- Use exterior BGP as the routing protocol between the fabric and the WAN router. In this case, we just exchange routes, not EVPN information.




## WAN Router Integration Using L3 Router Attach:

- This method does not work if you have disabled VRFs. You need at least one VRF.
- Between the WAN router and the service block functions you need a routing protocol to deal with failovers in case of a lost link. You can choose between OSPF or eBGP:
  - OSPF may be simpler to configure but needs some additional CLI for the added route filters.
  - eBGP allows you to configure everything in the GUI but it is a bit more complex. This is the Juniper recommended method of attaching a WAN router to a fabric.
- In the fabric dialogue there is no need to manually define additional routes per VRF since those will be obtained via OSPF or eBGP.
- For each VRF, select one of the existing attached VLANs to act as the uplink towards the WAN router as indirect mapping towards the VRF. In a production-grade, highly available environment, you have two WAN routers. You must have at least two VLAN's in each VRF before you can attach your WAN routers. This may be a bit strange but when referencing a VLAN that is bound to a VRF, the Mist UI knows how to reference the VRF. If you want, you can use device-transport VLANs per VRF, but you always need two as we expect a pair of redundant WAN routers for production.
- For each uplink VLAN (representing a VRF) you must have an IP subnet for L3 point-to-point (P2P) communication. Those subnets must be unique and non-overlapping with the pool the fabric uses

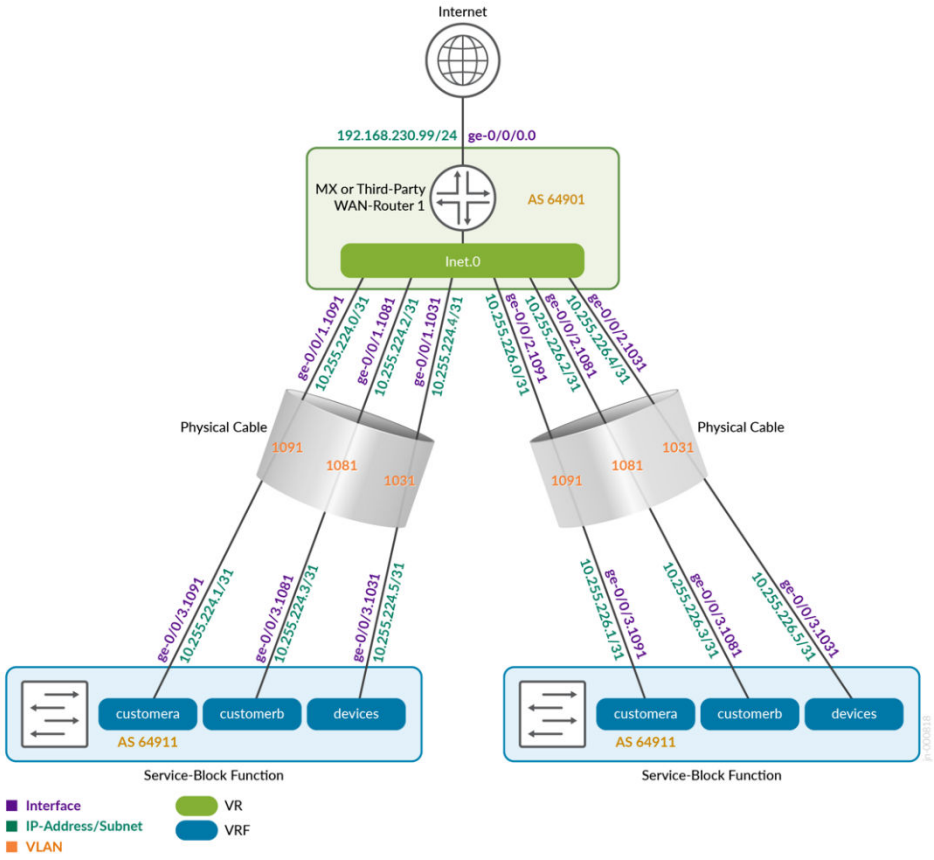
(usually 10.255.240.0/20). You can choose them on your own since you need to manage that assignment manually. It may seem strange that you overwrite the IP addresses of the fabric VLAN in the sub-interface definition, but this is by design. For the P2P links, we recommend using /31 networks outside of the fabric range mentioned above.

- While most of the configuration can be done in the Mist GUI, you must provide a few lines of additional CLI for OSPF:
  - This is to provide needed policy statements for the import and export filters for the OSPF area for each VRF. With eBGP you can manage those filters in the Mist GUI.
  - You also need to set a unique OSPF router ID for each VRF. This is to ensure that routes from the WAN router (such as the default route) are imported into each fabric VRF individually.
- When choosing eBGP you must manage your own private Autonomous System Numbers (ASN). The Mist fabric starts with the ASN 65000 so you must choose and ASN lower than that. We do not recommend the use of a unique AS per VRF because the maximum number of local ASN on a QFX swich is 16. We recommend that you use a shared ASN among your VRFs.

**NOTE**: All L3 methods per VRF and WAN router on the uplink for the P2P links are multiplexed into each link. In a production-grade design you would expect to have two WAN routers. This means that each VRF needs a minimum of two VLANs to make the connection to the outside.

Figure 4 on page 17 shows an example of an eBGP configuration for two service block functions and a single WAN router.

**Figure 4: eBGP Configuration for Two Service Block Functions and a Single WAN Router**




**NOTE**: We have provided configuration examples for all the WAN router attach methods in the appendix. If something is unclear, check the appendix.

# Validation Framework

**IN THIS SECTION**

## Test Bed

You are free to choose the integration either directly at the core switch or with a pair of dedicated service block switches. Therefore, we used the network architectures shown in Figure 5 on page 18 and in this JVDE.

**Figure 5: 5-Stage IP-Clos, ERB or CRB Fabric - No Dedicated Service Block**

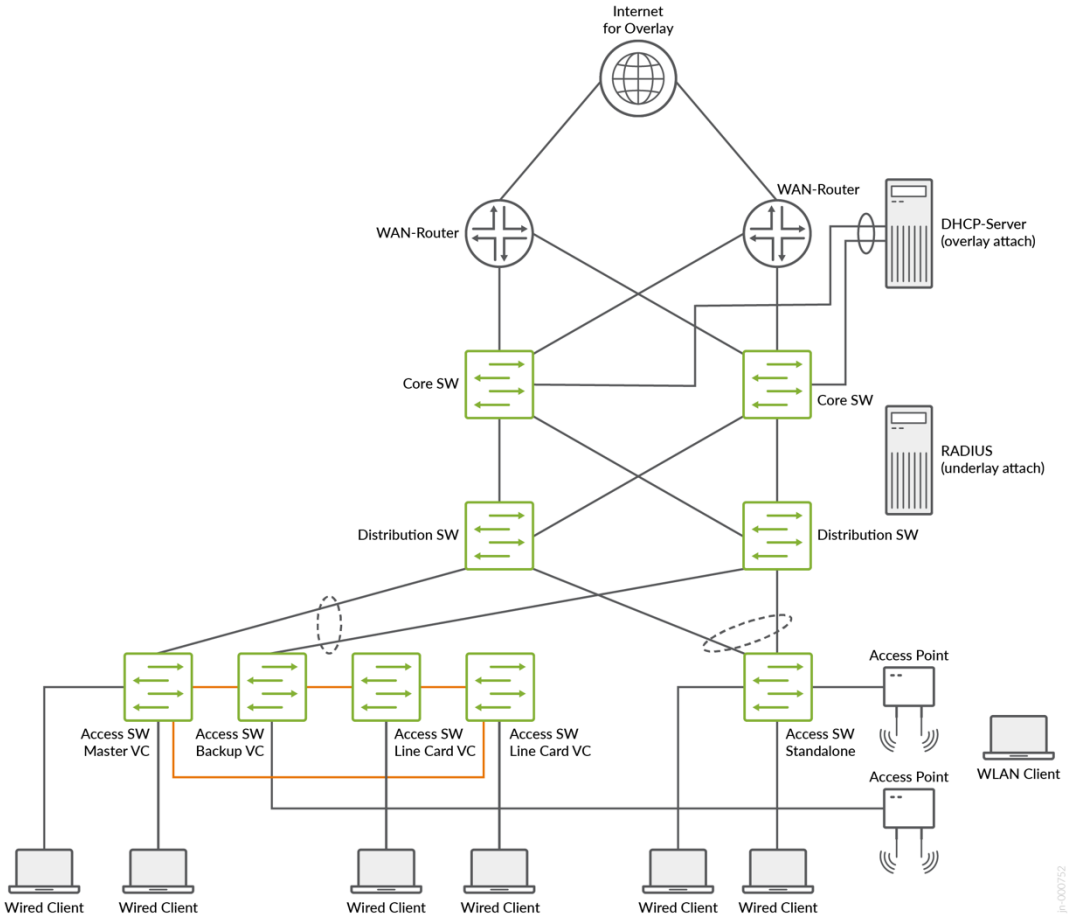
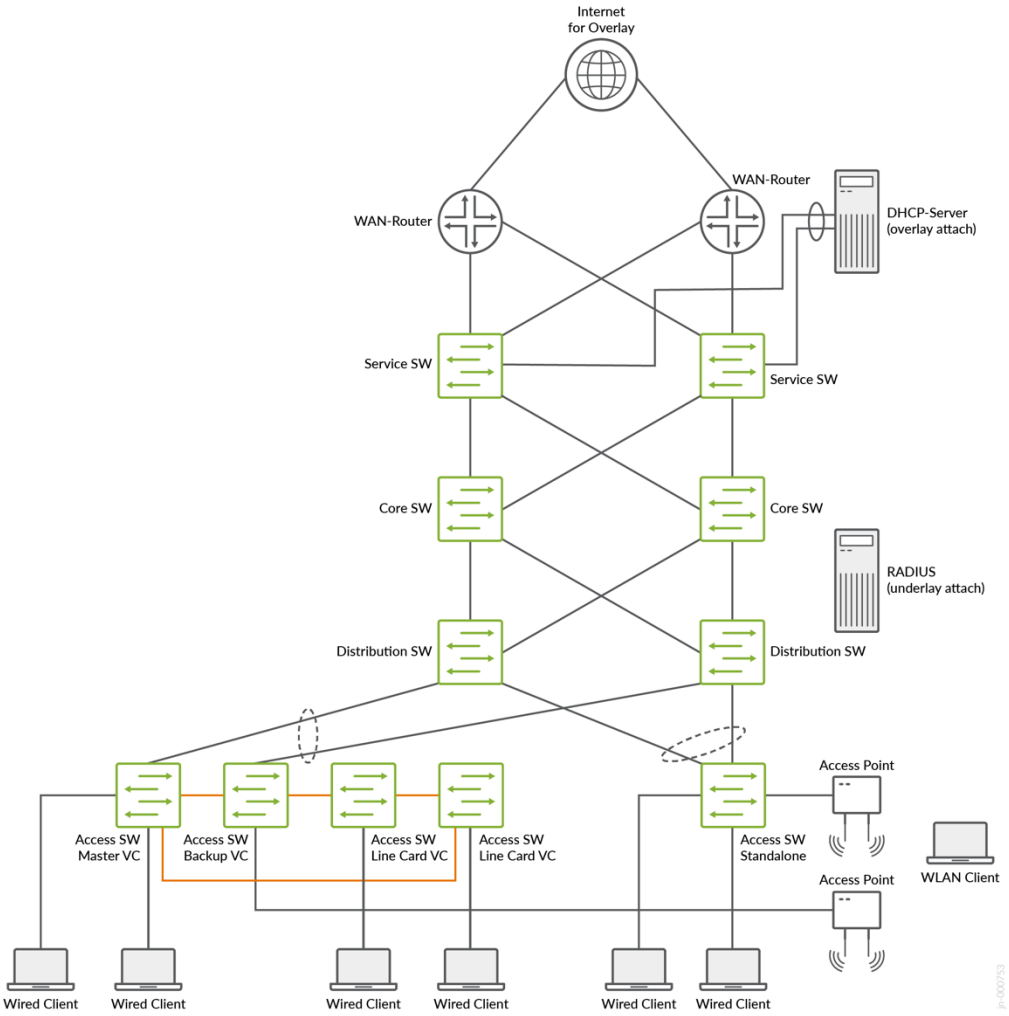


**Figure 6: 5-Stage IP-Clos, ERB, or CRB Fabric with Dedicated Service Blocks**




In the first case, the WAN routers directly attached to the core routers were Juniper MX routers. In the second case the WAN routers attached to the dedicated block of service switches were Juniper SRX Series Firewalls in HA cluster mode.

The fabric type can be either ERB or IP-Clos because in those Fabrics the Core router only performs a forwarding function. Due to our Mist GUI configuration, the L2 or L3 Junos configuration is added as needed so we can easily document the differences in the configurations.

## Platforms / Devices Under Test (DUT)

To review the software versions and platforms on which this JVD was validated by Juniper Networks, see the Validated Platforms and Software section in this document.

Testing was performed with a focus on the Juniper MX-Routers.

## Test Bed Configuration

In the appendix section, we share information about how some of the tests performed. Contact your Juniper representative to obtain the complete test bed configuration used for this JVDE.

# Test Objectives

**IN THIS SECTION**



## Test Goals

The testing for this JVDE was performed with the following goals in mind. Please also consult the Campus Fabric WAN Router Integration Test Report for more information.

The goals for this testing were:

- All tests must be performed with a redundant pair of WAN routers like you would deploy in a production-grade environment.
- Test the two major L2 attach functions for stretched VLAN and transport VLAN exit.
- Test the two L3 attach Functions for eBGP-based and OSPF-based exit.
- Test when the WAN router is attached to a core switch as virtual service block function.
- Test when the WAN router is attached to a pair of dedicated service block functions as a physical service block function.
- Test with the following Juniper Devices as WAN router:
  - Juniper MX Series router.

    - Juniper SRX Series Firewall in HA cluster mode.
- Perform failovers simulating a WAN router or service block function outage.
- Perform scale-out tests with 10 VRFs and 500 VLANs distributed in the fabric.

### Test Non-Goals

- The testing for this JVDE was not performed, for various reasons, on the following items:
- Bridged overlay was not tested since it is an option that should not be used in a production environment.
- No testing with any third-party WAN router or firewall was performed.
- Testing with redundant Ethernet (reth) interfaces on the SRX cluster was stopped during evaluation because the reth interfaces are not active/active. This restriction would require Mist to send additional CLI for full fabric configuration.

# Recommendations

The following summary lists again the recommendations that were given throughout the document.

- If you have free choice of WAN router type and attachment protocol, we recommend that you choose a WAN router such as a Juniper MX Series router and use eBGP as the L3 attachment protocol to the fabric. This is the most robust method and would likely support new features when they become available.
- We recommend the use of a pair of dedicated physical service block functions. In fact, this is a requirement if you have more than two core switches.
- For small fabrics, or lab and PoC designs, you can:
    - Use a single WAN router rather than a redundant pair of WAN routers.
    - Use any of the other tested methods such as stretched VLAN, transport VLAN, or OSPF.
- Feature requirements for third-party WAN routers to support L2 fabric attach:
    - IEEE 802.3ad LAG with active LACP. Without this, you must use an L3 attach method.
    - If the WAN router supports 2 or more devices for HA, it must support a failover mechanism such as VRRP for its GW IP.

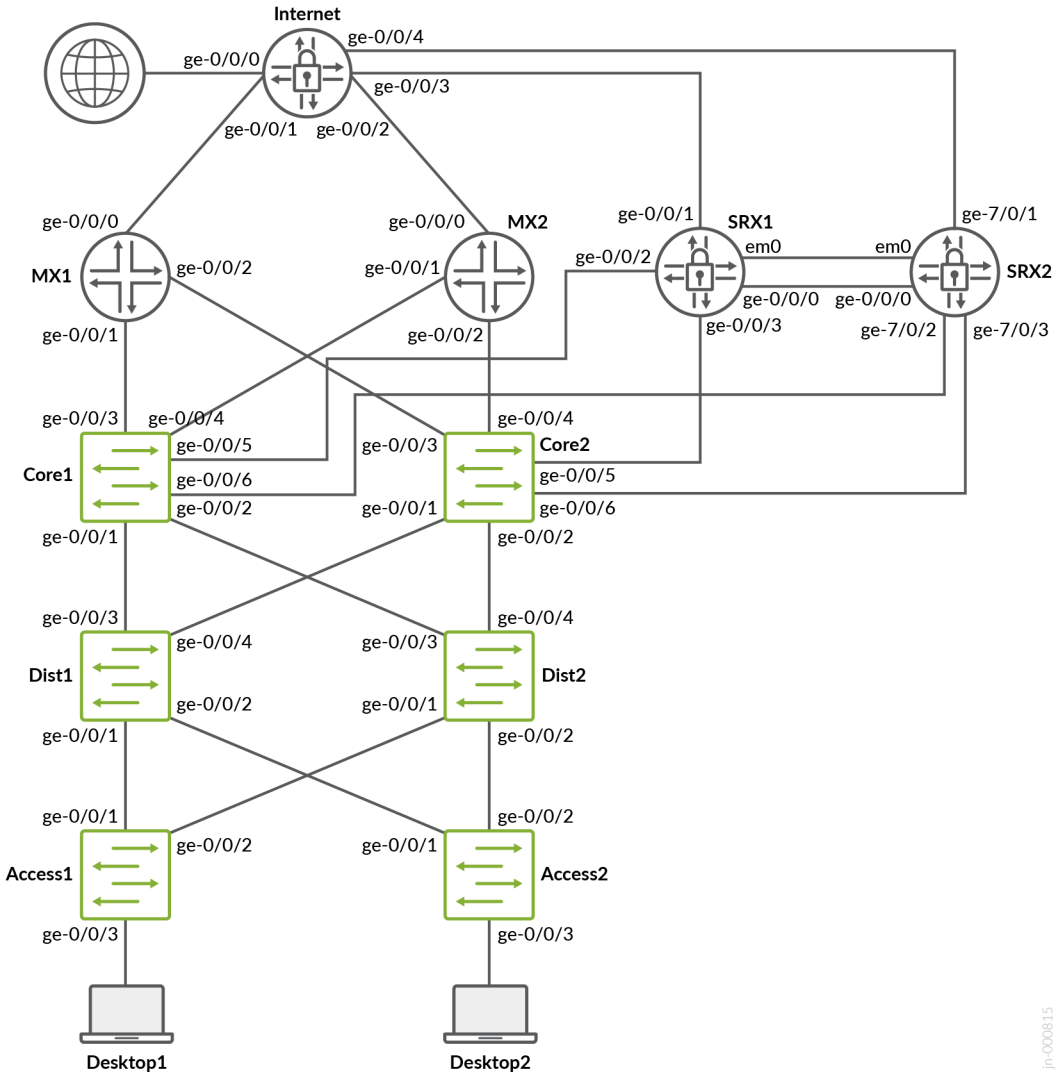
- Feature requirements for third-party WAN routers to support L3 fabric attach:
  - OSPF or standard eBGP-based route exchanges.
  - OSPF is easier and less to configure but currently lacks filters for import and export policies in the Mist GUI.
  - We recommend that you use eBGP with third-party WAN routers, but this requires more work to set up.
- Know and understand if the fabric you use is a virtual gateway fabric or an anycast fabric:
  - For virtual gateway fabrics you must leave room for four static IP addresses in each overlay VLAN that the fabric might use.
  - For anycast fabrics you must export host routes from the overlay loopback per-VRF subnet (typically 172.16.192.0/19) for future DHCP relay use.
- If you use a firewall as WAN router, make sure the vendor provides a way to synchronize the firewall states between the two devices. In addition, the failover mechanism provided must be stateful.
- When using Mist edges, they should be attached to a single service block function (one Mist edge per service block function). This helps to limit the MAC movement announcements through the fabric.
- When using an L2 attach with transport VLAN, make sure the netmask is /28 or longer. Also, remember to leave room for four static IPs for virtual gateways and the static IPs for VRRP on the WAN router side. We also recommend using virtual-gateway-address (VGA) configuration for optimal traffic forwarding in the fabric.
- When using bridged overlay, be aware of the limitations discussed previously.

# Appendix: Test Case Example Information

**IN THIS SECTION**

## Virtual Test Lab

The examples in this appendix to the JVDE are evaluated in a virtual test lab consisting of a vJunos-switch , a vMX router, and vSRX V3.0 firewalls. We did not create a pair of virtual service block switches but ensured that both types of WAN routers (router or firewall) were available as redundant pairs. This is not the same lab used for testing that required the use physical hardware. We use this as an example and something you can potentially build yourself with environments such as EVE-NG to build your own labs to test the configuration examples. The fabric, in this case, was configured as IP Clos.




## L2 Exit with Stretched VLAN

**NOTE**: Previous versions of this JVD-E included an example configuration for L2 Exit using stretched VLANs. However, this led to significant issues, as users often overlooked the fact that it was never intended for production use. To avoid further confusion, we have decided to remove the example entirely. While it may still appear in older documentation, it remains an unsupported and non-production-grade solution.

## L2 Exit with Transport VLAN

**NOTE**: When doing any VLAN or VRF creation with campus fabric remember the following best practices:Create all VLANs in a switch template and then import them in the **Campus Fabric** dialogue. Creating the VLANs anywhere else in the Mist GUI ultimately leads to inconsistency which makes it hard to resolve issues.With the exception of the service block functions, do not create VRFs outside of the **Campus Fabric** dialogue.The transport VLAN method requires you to create VRFs manually on the service block function and add the transport VLAN and routes locally to the VRFs. Do not create the VRFs or routes in the **Campus Fabric** dialogue.We recommend that you create port profiles within switch templates so that any changes are in sync on all switches in the fabric.

When defining the transport VLANs in the switch template, do not set the subnet information. You configure this information later as **Additional IP Subnet** on each service block function. See Figure 7 on page 25 , Figure 8 on page 26 , and Figure 9 on page 26 .

**Figure 7: Empty Subnet Configuration on Transport VLAN 1**

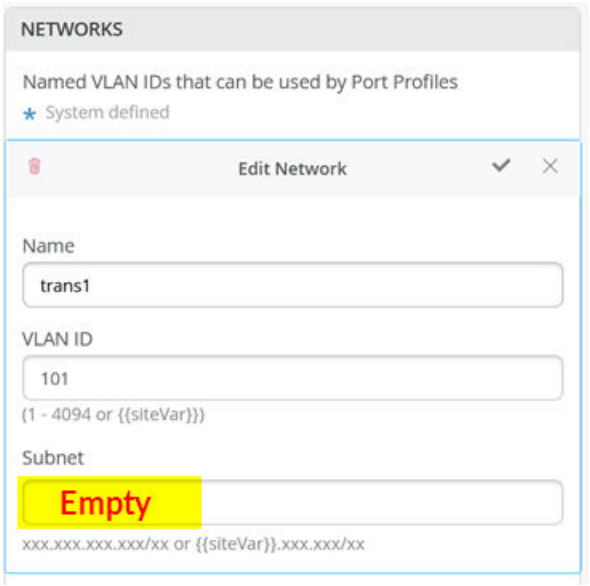


**Figure 8: Empty Subnet Configuration on Transport VLAN 2**

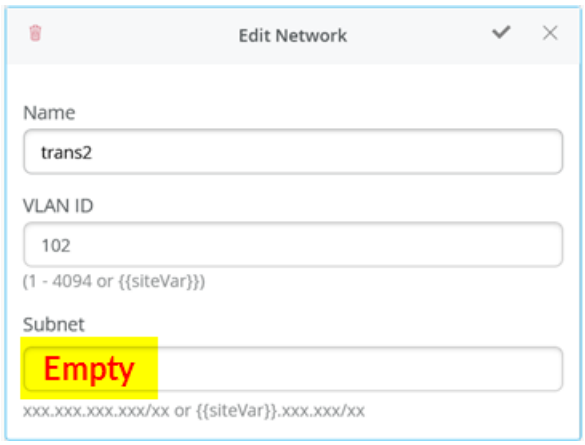
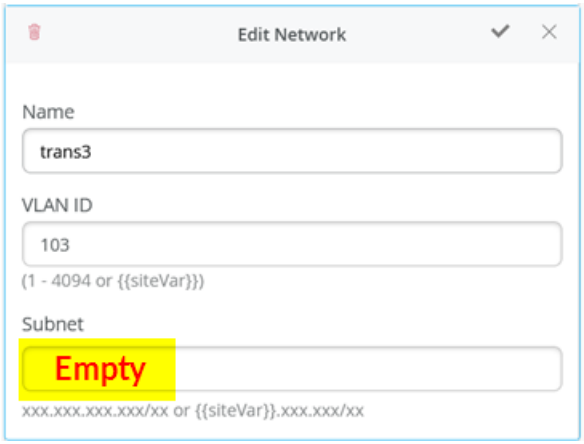



**Figure 9: Empty Subnet Configuration on Transport VLAN 3**




The following CLI configuration shows the exported version of the switch template used in the transport VLAN fabric. This allows you to review our setup when importing. As you can see, there is a minimum of two VLANs per VRF plus an additional transport VLAN per VRF.

```
{
  “additional_config_cmds”: [],
  “networks”: {
    “vlan1099”: {
      “vlan_id”: 1099,
      “subnet”: “10.99.99.0/24”
    },
    “vlan1088”: {
      “vlan_id”: 1088,
```

```
      "subnet": "10.88.88.0/24"
    },
    "vlan1033": {
      "vlan_id": 1033,
      "subnet": "10.33.33.0/24"
    },
    "vlan1091": {
      "vlan_id": 1091,
      "subnet": "10.99.91.0/24"
    },
    "vlan1081": {
      "vlan_id": 1081,
      "subnet": "10.88.81.0/24"
    },
    "vlan1031": {
      "vlan_id": 1031,
      "subnet": "10.33.31.0/24"
    },
    "trans1": {
      "vlan_id": "101",
      "subnet": ""
    },
    "trans2": {
      "vlan_id": "102",
      "subnet": ""
    },
    "trans3": {
      "vlan_id": "103",
      "subnet": ""
    }
  },
  "port_usages": {
    "vlan1099": {
      "mode": "access",
      "disabled": false,
      "port_network": "vlan1099",
      "voip_network": null,
      "stp_edge": false,
      "mac_auth_protocol": null,
      "all_networks": false,
      "networks": null,
      "port_auth": null,
      "enable_mac_auth": null,
```

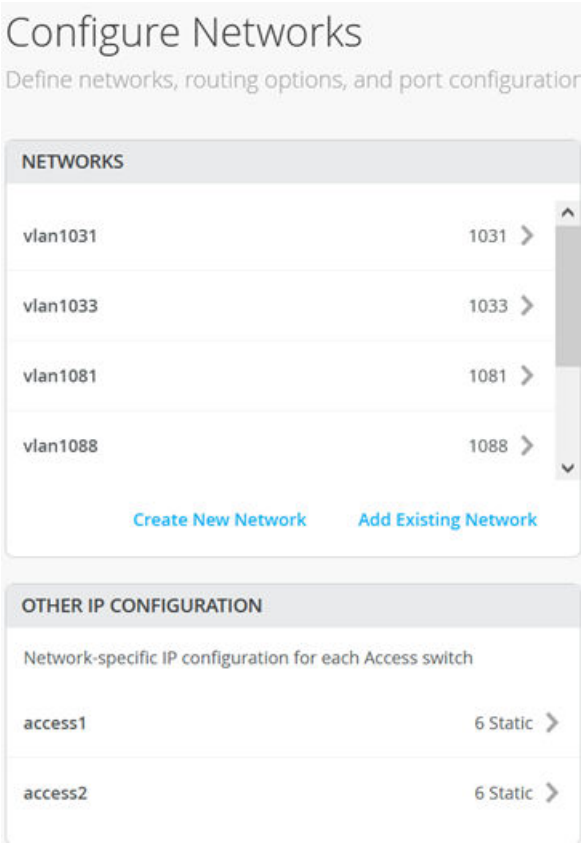
```
      “mac_auth_only”: null,
      “guest_network”: null,
      “bypass_auth_when_server_down”: null,
      “speed”: “auto”,
      “duplex”: “auto”,
      “mac_limit”: 0,
      “persist_mac”: false,
      “poe_disabled”: false,
      “enable_qos”: false,
      “storm_control”: {},
      “mtu”: null,
      “description”: “”,
      “disable_autoneg”: false
    },
    “vlan1088”: {
      “mode”: “access”,
      “disabled”: false,
      “port_network”: “vlan1088”,
      “voip_network”: null,
      “stp_edge”: false,
      “mac_auth_protocol”: null,
      “all_networks”: false,
      “networks”: null,
      “port_auth”: null,
      “enable_mac_auth”: null,
      “mac_auth_only”: null,
      “guest_network”: null,
      “bypass_auth_when_server_down”: null,
      “speed”: “auto”,
      “duplex”: “auto”,
      “mac_limit”: 0,
      “persist_mac”: false,
      “poe_disabled”: false,
      “enable_qos”: false,
      “storm_control”: {},
      “mtu”: null,
      “description”: “”,
      “disable_autoneg”: false
    },
    “dynamic”: {
      “mode”: “dynamic”,
      “rules”: []
    }
```

```
  },
  “switch_matching”: {
    “enable”: true,
    “rules”: []
  },
  “switch_mgmt”: {
    “config_revert_timer”: 10,
    “root_password”: “<password>”,
    “protect_re”: {
      “enabled”: false
    },
    “tacacs”: {
      “enabled”: false
    }
  },
  “mist_nac”: {
    “enabled”: true,
    “network”: null
  },
  “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
  },
  “dns_servers”: [],
```

```
  “dns_suffix”: [],
  “ntp_servers”: [],
  “acl_policies”: [],
  “port_mirroring”: {},
  “name”: “campus-fabric”
}
```

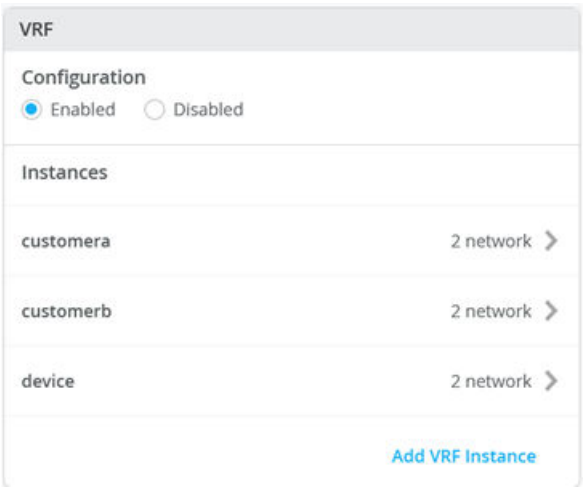
Within the **Campus Fabric Configuration** dialogue, there is a section called **Configure Networks**. This is where you import your six access VLANs from the switch template. When finished, the configuration should be as shown in Figure 10 on page 30 and the result in our case will look as shown below. Since the three transport VLANs are not part of the access layer, they are not defined in the service block function.

**Figure 10: Access VLAN Import Within Campus Fabric Configuration Dialogue**




Next, you create 3 VRFs and attach two of the access networks to each VRF as shown in Figure 11 on page 31 .

**Figure 11: VRF Configuration**




Next, go to each VRF and confirm that you only have access networks defined with no default route. You will define the transport VLANs and default routes later in the service block function. See Figure 12 on page 31, Figure 13 on page 32 , and Figure 14 on page 32 .

**Figure 12: VRF1—Access VLANs Without Default Routes**

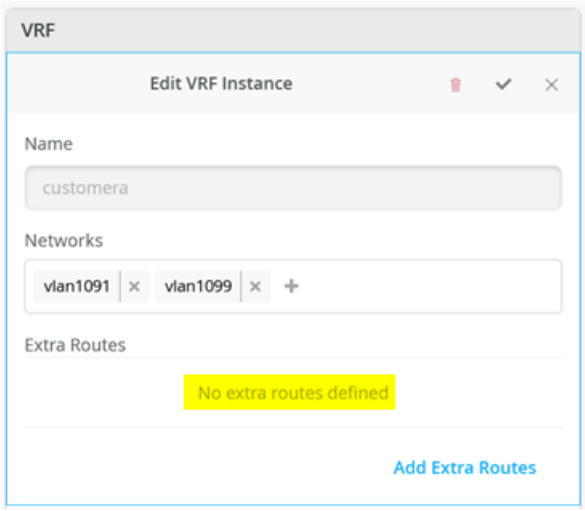


**Figure 13: VRF2—Access VLANs Without Default Routes**

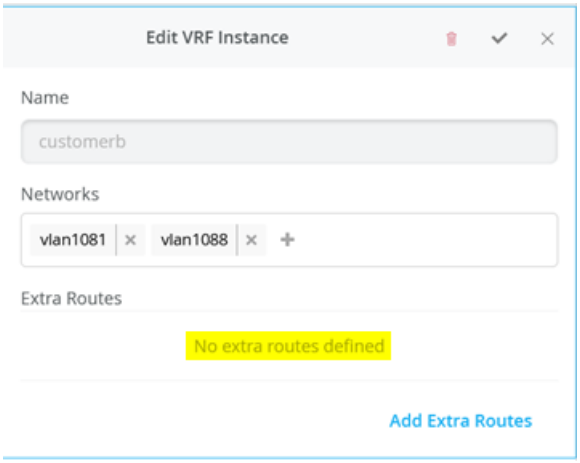
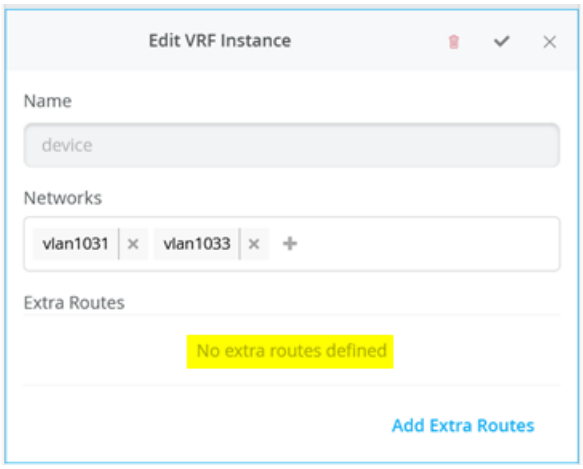



**Figure 14: VRF3—Access VLANs Without Default Routes**




### Core1 and Core2 Switch Configuration

In the transport VLAN attach example, the service block function is virtual and co-located on the core switch. Therefore, you must configure the two core switches. The following pseudocode represents the configuration you must apply to the core1 and core2 switches:

```
# configure the additional IP subnet 10.99.1.1/28 to network/VLAN:trans1
# configure the additional IP subnet 10.88.1.1/28 to network/VLAN:trans2
# configure the additional IP subnet 10.33.1.1/28 to network/VLAN:trans3
#
# Create a new local Port Profile called 'l2fabricexit' and configure:
# Mode='Trunk'
```

```
# Port Network (Untagged/Native VLAN)='None'
# Add the following 3 Networks as Trunk Networks:
# Network=trans1
# Network=trans2
# Network=trans3
# MTU='9018'
#
# Create a new Port configuration where:
# Port IDs=ge-0/0/3
# Interface=L2 Interface
# Configuration Profile=l2fabricexit
# Port Aggregation=Enable/Checked
# AE Index=11
# ESI-LAG=Enable/Checked
#
# Create a new Port configuration where:
# Port IDs=ge-0/0/4
# Interface=L2 Interface
# Configuration Profile=l2fabricexit
# Port Aggregation=Enable/Checked
# AE Index=12
# ESI-LAG=Enable/Checked
#
# In VRF Configuration
# Override Site/Template Settings=Checked
# In Instance customera
# Override Template Defined VRF Instance=Checked
# Add the Network trans1 to the existing list of networks
# Add the Extra Route 0.0.0.0/0 with via: 10.99.1.14
#
# In Instance customerb
# Override Template Defined VRF Instance=Checked
# Add the Network trans2 to the existing list of networks
# Add the Extra Route 0.0.0.0/0 with via: 10.88.1.14
#
# In Instance device
# Override Template Defined VRF Instance=Checked
# Add the Network trans3 to the existing list of networks
# Add the Extra Route 0.0.0.0/0 with via: 10.33.1.14
```

The following four images display the Mist GUI configuration that results from the previous pseudocode starting with the additional IP configuration required to assign the local IP addresses to each transport VLAN.

**Figure 15: Transport VLAN Additional IP Configuration**

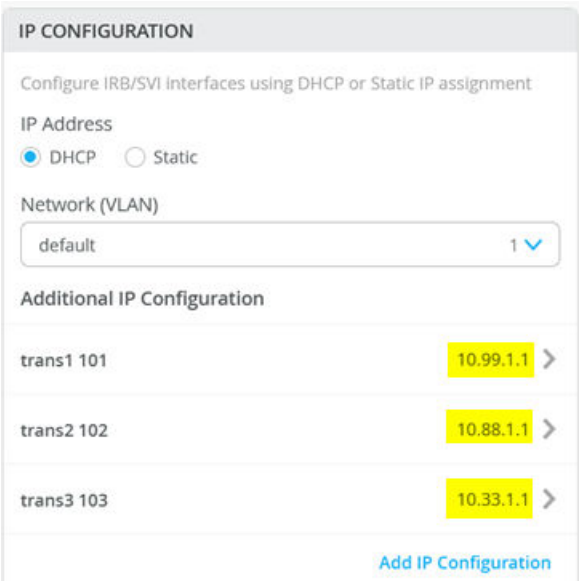



**Figure 16: VLAN trans1 Configuration**

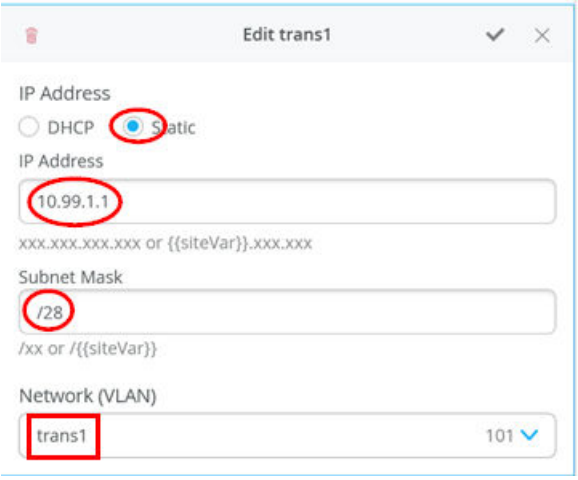


**Figure 17: VLAN trans2 Configuration**

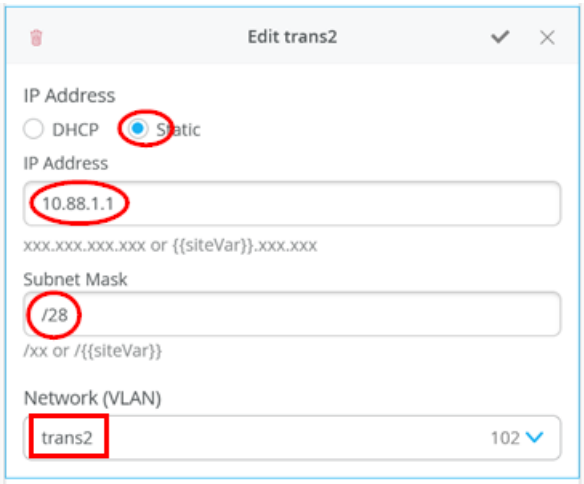
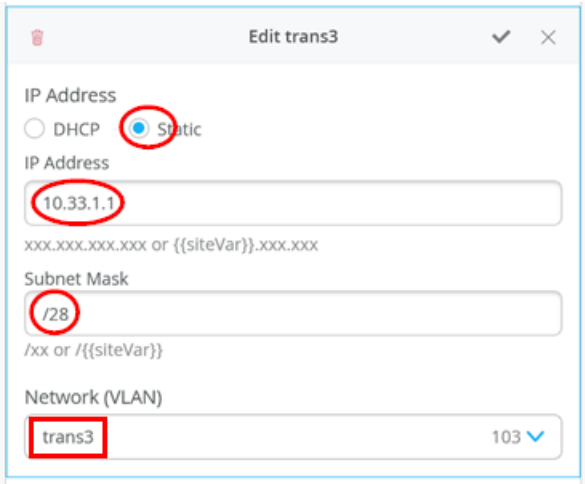
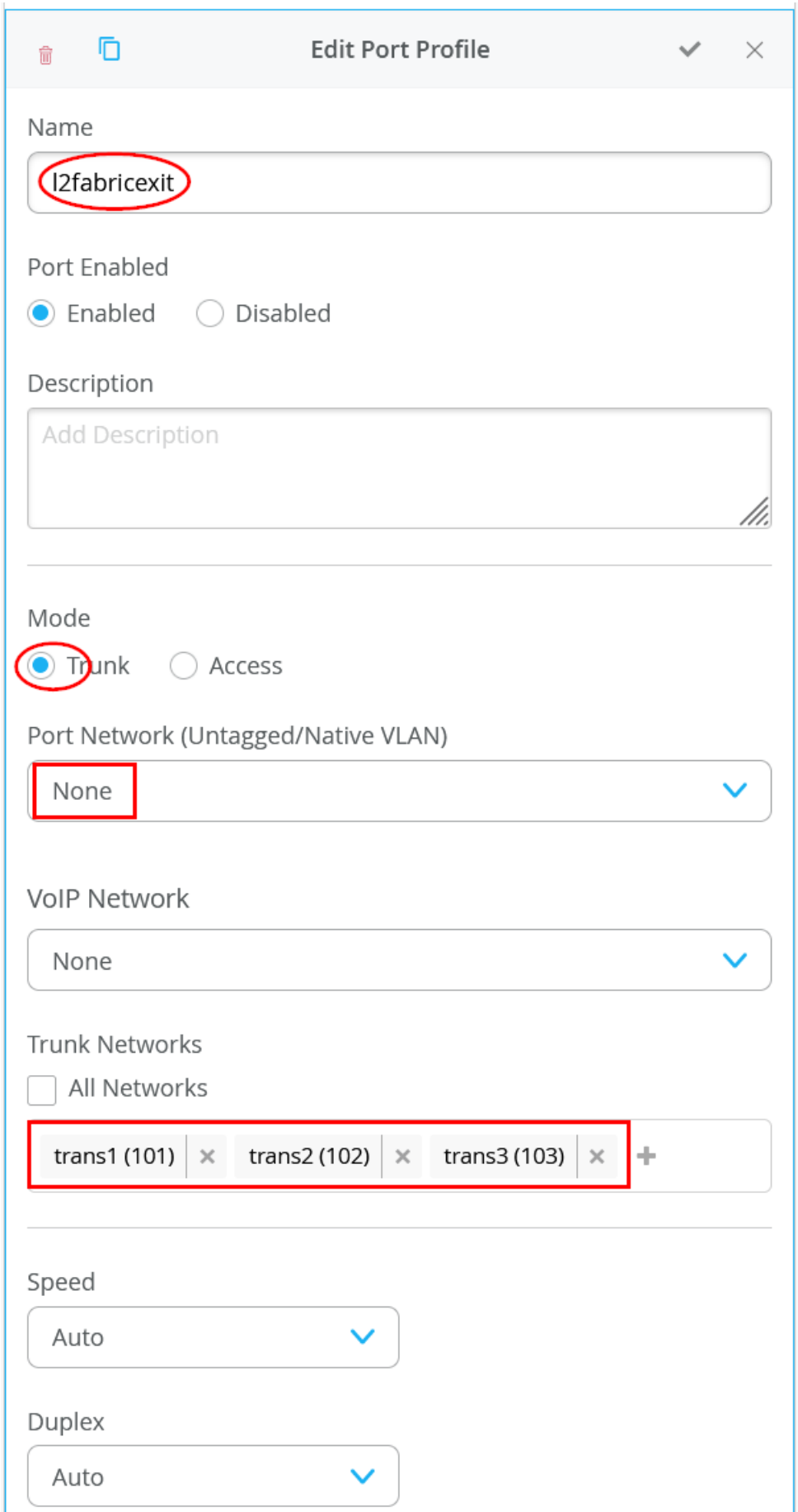



**Figure 18: VLAN trans3 Configuration**




Next, you define the **Port Profile** used for the uplinks. It is critical that you only include the transport VLAN in the Trunk Networks definition since only those VLANs are used and visible to the WAN router.

**Figure 19: Port Profile for WAN Router Attach Using Transport VLAN**



Next, you assign the port profiles to each uplink port.

**Figure 20: Port Profile Assignment for Transport VLAN Attach**

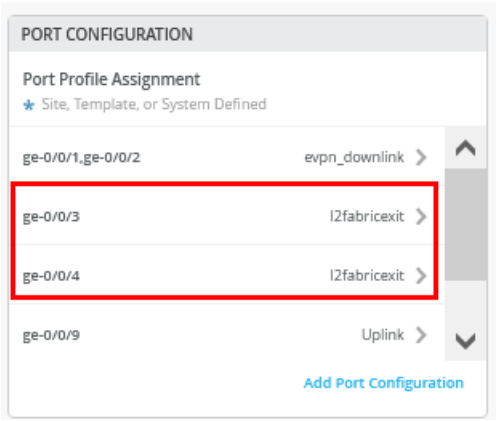



Figure 21 on page 38 shows the configuration of the first uplink to the first WAN router.

**Figure 21: Port Configuration for First Uplink to First WAN Router**

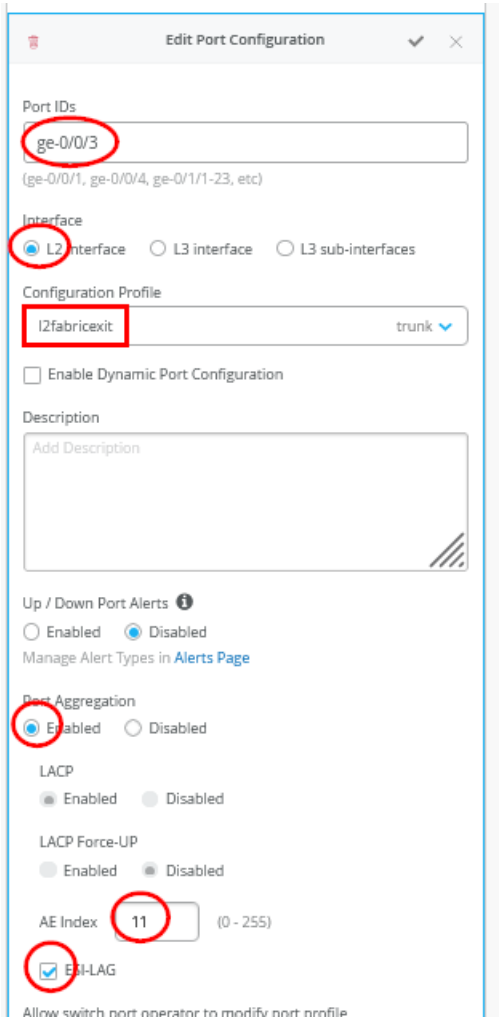
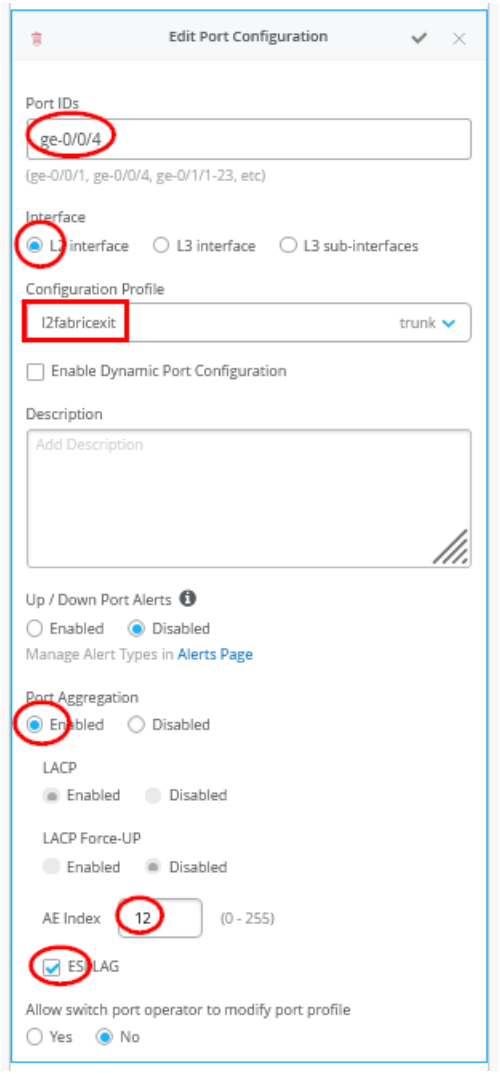



Figure 22 on page 39 shows the configuration of the second uplink to the first WAN router.

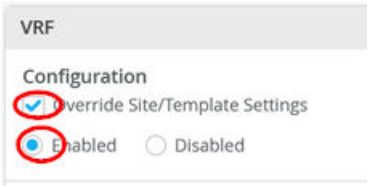
**Figure 22: Port Configuration for Second Uplink to First WAN Router**




**NOTE**: You must ensure that the **AE Indexes** on each service block function are in sync with each other towards the same WAN router and that you define them each as **ESI-LAG**. You must also ensure that you don't reuse an AE Index that is already defined elsewhere in the fabric service block.

Next you create and modify local VRFs. Remember this is an exception made only for the transport VLAN exit method. Usually, the fabric creates the VRFs automatically. In this case we must enable the **Override Site/Template Settings** checkbox in the VRF configuration. Figure 23 on page 40 shows the required configuration in the Mist GUI.

**Figure 23: Override Template Settings for Transport VLAN Exit**




Next you must perform the following three configurations in each of your three VRS instances:

- Enable **Override Template Defined VRF Instance** checkbox
- Add your transport VLAN to the pre-populated list of access VLANs
- Add a default route where the gateway IP address is the VRRP-VIP address of your WAN router.
- Figure 24 on page 40 , Figure 25 on page 41 , and Figure 26 on page 41 show the override configurations for each of the three VRFs.

**Figure 24: VRF1 Override Configurations**

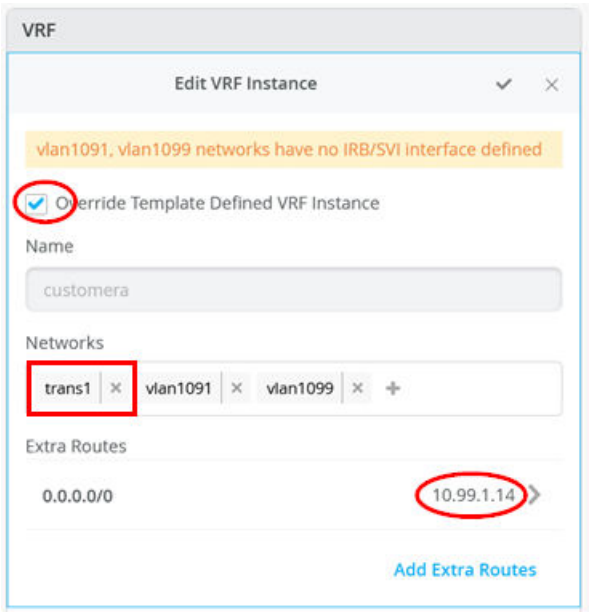


**Figure 25: VRF2 Override Configuration**

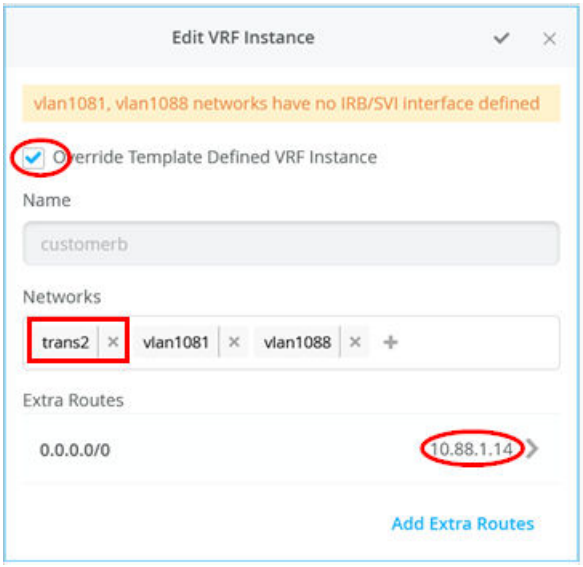
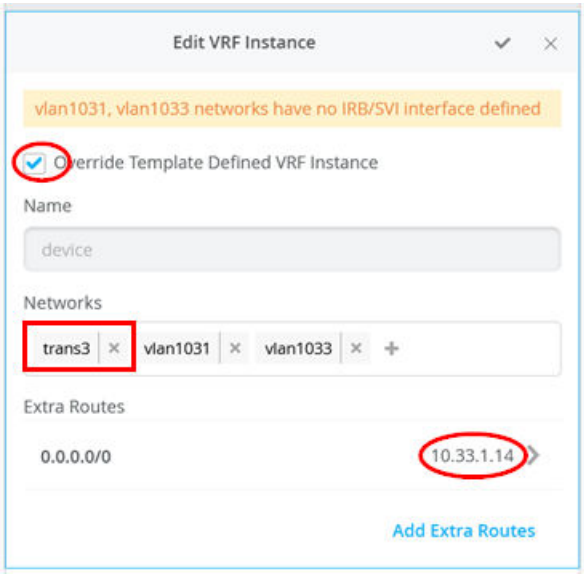
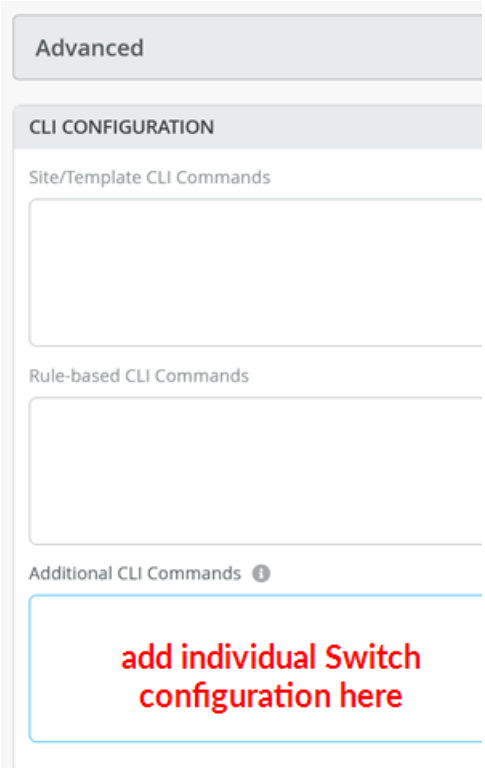



**Figure 26: VRF3 Override Configuration**




Now you must configure additional CLI to modify the transport VLANs to use VGA configuration to help avoid excess hair-pin routing of traffic within the fabric. In the switch configuration for each of your service block function switches, locate the **CLI Configuration** section in the Mist GUI. You must paste the required configuration into the field indicated in Figure 27 on page 42.

**Figure 27: Location of Additional CLI Commands Field**




The example CLI configuration for your **core1 switch,** is shown in the following code block. We have configured the static IP address as the virtual gateway IP address + 1 (10.99.1.2).

```
# when service block function is a EX92xx change to VGA with the below
delete groups top routing-instances evpn_vs protocols evpn default-gateway do-not-advertise
set groups top routing-instances evpn_vs protocols evpn default-gateway no-gateway-community
#
# on non-EX92xx switches change to VGA with the below
# delete groups top protocols evpn default-gateway do-not-advertise
# set groups top protocols evpn default-gateway no-gateway-community
#
# modify our transport VLANs to VGA
delete interfaces irb unit 101 family inet address 10.99.1.1/28
set interfaces irb unit 101 family inet address 10.99.1.2/28 virtual-gateway-address 10.99.1.1
set interfaces irb unit 101 virtual-gateway-accept-data
set interfaces irb unit 101 virtual-gateway-v4-mac 00:00:5e:e4:05:01
#
delete interfaces irb unit 102 family inet address 10.88.1.1/28
set interfaces irb unit 102 family inet address 10.88.1.2/28 virtual-gateway-address 10.88.1.1
set interfaces irb unit 102 virtual-gateway-accept-data
```

```
set interfaces irb unit 102 virtual-gateway-v4-mac 00:00:5e:e4:05:02
#
delete interfaces irb unit 103 family inet address 10.33.1.1/28
set interfaces irb unit 103 family inet address 10.33.1.2/28 virtual-gateway-address 10.33.1.1
set interfaces irb unit 103 virtual-gateway-accept-data
set interfaces irb unit 103 virtual-gateway-v4-mac 00:00:5e:e4:05:03
```

For your **core2 switch,** only the static IP addresses of the transport VLAN are changed to be the virtual gateway IP address + 2 (10.88.1.3).

```
# when service block function is a EX92xx change to VGA with the below
delete groups top routing-instances evpn_vs protocols evpn default-gateway do-not-advertise
set groups top routing-instances evpn_vs protocols evpn default-gateway no-gateway-community
#
# on all non-EX92xx switches change to VGA with the below
# delete groups top protocols evpn default-gateway do-not-advertise
# set groups top protocols evpn default-gateway no-gateway-community
#
# modify our transport VLANs to VGA
delete interfaces irb unit 101 family inet address 10.99.1.1/28
set interfaces irb unit 101 family inet address 10.99.1.3/28 virtual-gateway-address 10.99.1.1
set interfaces irb unit 101 virtual-gateway-accept-data
set interfaces irb unit 101 virtual-gateway-v4-mac 00:00:5e:e4:05:01
#
delete interfaces irb unit 102 family inet address 10.88.1.1/28
set interfaces irb unit 102 family inet address 10.88.1.3/28 virtual-gateway-address 10.88.1.1
set interfaces irb unit 102 virtual-gateway-accept-data
set interfaces irb unit 102 virtual-gateway-v4-mac 00:00:5e:e4:05:02
#
#
delete interfaces irb unit 103 family inet address 10.33.1.1/28
set interfaces irb unit 103 family inet address 10.33.1.3/28 virtual-gateway-address 10.33.1.1
set interfaces irb unit 103 virtual-gateway-accept-data
set interfaces irb unit 103 virtual-gateway-v4-mac 00:00:5e:e4:05:03
```

**NOTE**: Keep in mind that our test lab used virtual EX9214 switches as core switches. In most production environments you will not use an EX92xx switch. Therefore, you must uncomment the two lines that are commented out in the previous configuration snippet:

```
# delete groups top protocols evpn default-gateway do-not-advertise
# set groups top protocols evpn default-gateway no-gateway-community
```

**NOTE**: Service block for each transport VLAN used per VRF you must manually set the MAC address of the virtual gateway address used on the IRB interface. In our example, we used different MAC addresses per transport VLAN because it's easier to debug.

## Juniper MX as WAN Router

The following CLI snippet example contains the configuration of the interfaces, the VRRP gateway redundancy, and the static routes for the first WAN router. You may need to add default routes and interfaces to complete the configuration.

```
set system host-name wanrouter1
#
set chassis aggregated-devices ethernet device-count 10
#
delete interfaces ae0
delete policy-options policy-statement fabric
delete policy-options policy-statement internet
delete routing-instances public-int
#
set protocols lldp port-id-subtype interface-name
set protocols lldp interface all
set protocols lldp-med interface all
#
delete interfaces ge-0/0/1
set interfaces ge-0/0/1 gigether-options 802.3ad ae11
delete interfaces ge-0/0/2
set interfaces ge-0/0/2 gigether-options 802.3ad ae11
```

```
delete interfaces ae11
set interfaces ae11 mtu 9018
set interfaces ae11 aggregated-ether-options lacp active
set interfaces ae11 aggregated-ether-options lacp admin-key 11
set interfaces ae11 unit 0 family bridge interface-mode trunk
set interfaces ae11 unit 0 family bridge vlan-id-list 101
set interfaces ae11 unit 0 family bridge vlan-id-list 102
set interfaces ae11 unit 0 family bridge vlan-id-list 103
#
set bridge-domains trans1 vlan-id 101
set bridge-domains trans1 routing-interface irb.101
set bridge-domains trans2 vlan-id 102
set bridge-domains trans2 routing-interface irb.102
set bridge-domains trans3 vlan-id 103
set bridge-domains trans3 routing-interface irb.103
#
set interfaces irb unit 101 family inet address 10.99.1.13/28 vrrp-group 1 virtual-address
10.99.1.14
set interfaces irb unit 101 family inet address 10.99.1.13/28 vrrp-group 1 priority 110
set interfaces irb unit 101 family inet address 10.99.1.13/28 vrrp-group 1 accept-data
#
set interfaces irb unit 102 family inet address 10.88.1.13/28 vrrp-group 2 virtual-address
10.88.1.14
set interfaces irb unit 102 family inet address 10.88.1.13/28 vrrp-group 2 priority 110
set interfaces irb unit 102 family inet address 10.88.1.13/28 vrrp-group 2 accept-data
#
set interfaces irb unit 103 family inet address 10.33.1.13/28 vrrp-group 3 virtual-address
10.33.1.14
set interfaces irb unit 103 family inet address 10.33.1.13/28 vrrp-group 3 priority 110
set interfaces irb unit 103 family inet address 10.33.1.13/28 vrrp-group 3 accept-data
#
set routing-options static route 10.99.91.0/24 next-hop 10.99.1.1
set routing-options static route 10.99.99.0/24 next-hop 10.99.1.1
set routing-options static route 172.16.193.0/24 next-hop 10.99.1.1
#
set routing-options static route 10.88.81.0/24 next-hop 10.88.1.1
set routing-options static route 10.88.88.0/24 next-hop 10.88.1.1
set routing-options static route 172.16.194.0/24 next-hop 10.88.1.1
#
set routing-options static route 10.33.31.0/24 next-hop 10.33.1.1
set routing-options static route 10.33.33.0/24 next-hop 10.33.1.1
set routing-options static route 172.16.195.0/24 next-hop 10.33.1.1
```

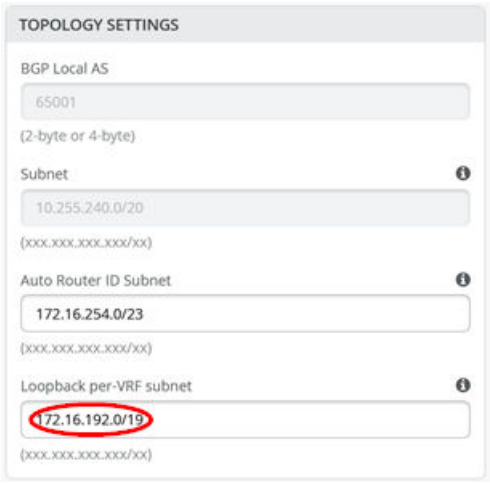
On the second WAN router, the notable configuration changes are the AE keys and indexes, and the static IP addresses.

```
set system host-name wanrouter2
#
set chassis aggregated-devices ethernet device-count 10
#
set protocols lldp port-id-subtype interface-name
set protocols lldp interface all
set protocols lldp-med interface all
#
delete interfaces ge-0/0/1
set interfaces ge-0/0/1 gigether-options 802.3ad ae12
delete interfaces ge-0/0/2
set interfaces ge-0/0/2 gigether-options 802.3ad ae12
#
delete interfaces ae12
set interfaces ae12 mtu 9018
set interfaces ae12 aggregated-ether-options lacp active
set interfaces ae12 aggregated-ether-options lacp admin-key 12
set interfaces ae12 unit 0 family bridge interface-mode trunk
set interfaces ae12 unit 0 family bridge vlan-id-list 101
set interfaces ae12 unit 0 family bridge vlan-id-list 102
set interfaces ae12 unit 0 family bridge vlan-id-list 103
#
set bridge-domains trans1 vlan-id 101
set bridge-domains trans1 routing-interface irb.101
set bridge-domains trans2 vlan-id 102
set bridge-domains trans2 routing-interface irb.102
set bridge-domains trans3 vlan-id 103
set bridge-domains trans3 routing-interface irb.103
#
set interfaces irb unit 101 family inet address 10.99.1.12/28 vrrp-group 1 virtual-address
10.99.1.14
set interfaces irb unit 101 family inet address 10.99.1.12/28 vrrp-group 1 accept-data
#
set interfaces irb unit 102 family inet address 10.88.1.12/28 vrrp-group 2 virtual-address
10.88.1.14
set interfaces irb unit 102 family inet address 10.88.1.12/28 vrrp-group 2 accept-data
#
set interfaces irb unit 103 family inet address 10.33.1.12/28 vrrp-group 3 virtual-address
10.33.1.14
```

```
set interfaces irb unit 103 family inet address 10.33.1.12/28 vrrp-group 3 accept-data
#
set routing-options static route 10.99.91.0/24 next-hop 10.99.1.1
set routing-options static route 10.99.99.0/24 next-hop 10.99.1.1
set routing-options static route 172.16.193.0/24 next-hop 10.99.1.1
#
set routing-options static route 10.88.81.0/24 next-hop 10.88.1.1
set routing-options static route 10.88.88.0/24 next-hop 10.88.1.1
set routing-options static route 172.16.194.0/24 next-hop 10.88.1.1
#
set routing-options static route 10.33.31.0/24 next-hop 10.33.1.1
set routing-options static route 10.33.33.0/24 next-hop 10.33.1.1
set routing-options static route 172.16.195.0/24 next-hop 10.33.1.1
```

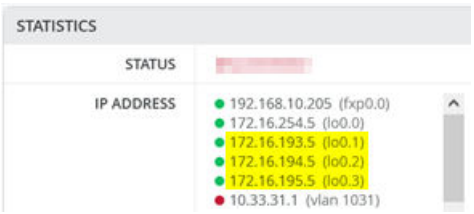
You may wonder about those static routes in the 172.16.19x.0 range. Remember that IP Clos is an anycast fabric. As such, you must have th static routes to prepare for when the DHCP relay will use IP addresses in the fabric overlay. See Figure 28 on page 47 for an example definition

**Figure 28: Loopback per VRF Subnet**




The overlay Loopbacks IPs are assigned to each VRF on a switch as a /24 range. You can figure them out by looking at a fabric access switch as shown in Figure 29 on page 48 . Hence, you must map them back like any other additional VLAN attached to the VRF to achieve the required reachability.

**Figure 29: VRF Loopback IP Addresses**




The following commands help to debug the connections on WAN router1.

```
root@wanrouter1> show lldp neighbors
Local Interface    Parent Interface    Chassis Id                              Port
info          System Name
ge-0/0/0           -                   4c:96:14:95:09:80
516                 internet
ge-0/0/1           ae11                2c:6b:f5:3a:42:c0
ge-0/0/3            core1
ge-0/0/2           ae11                2c:6b:f5:7f:7d:c0
ge-0/0/3            core2
.
root@wanrouter1> show lacp interfaces
Aggregated interface: ae11
    LACP state:           Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      ge-0/0/1          Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      ge-0/0/1        Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      ge-0/0/2          Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      ge-0/0/2        Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      ge-0/0/1                  Current   Fast periodic Collecting distributing
      ge-0/0/2                  Current   Fast periodic Collecting distributing
.
root@wanrouter1> show vrrp
Interface     State       Group   VR state VR Mode   Timer    Type   Address
irb.101       up              1   master   Active      A  0.350 lcl    10.99.1.13
                                                               vip    10.99.1.14
irb.102       up              2   master   Active      A  0.625 lcl    10.88.1.13
                                                               vip    10.88.1.14
irb.103       up              3   master   Active      A  0.830 lcl    10.33.1.13
                                                               vip    10.33.1.14
```

The following commands help you to debug connections on WAN router2.

```
root@wanrouter2> show lldp neighbors
Local Interface    Parent Interface    Chassis Id                              Port
info          System Name
ge-0/0/0           -                   4c:96:14:95:09:80
517                 internet
ge-0/0/1           ae12                2c:6b:f5:3a:42:c0
ge-0/0/4            core1
ge-0/0/2           ae12                2c:6b:f5:7f:7d:c0
ge-0/0/4            core2
.
root@wanrouter2> show lacp interfaces
Aggregated interface: ae12
    LACP state:           Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      ge-0/0/1          Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      ge-0/0/1        Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      ge-0/0/2          Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      ge-0/0/2        Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      ge-0/0/1                  Current   Fast periodic Collecting distributing
      ge-0/0/2                  Current   Fast periodic Collecting distributing
.
root@wanrouter2> show vrrp
Interface     State       Group   VR state VR Mode   Timer    Type   Address
irb.101       up              1   backup   Active      D  2.811 lcl    10.99.1.12
                                                              vip    10.99.1.14
                                                              mas    10.99.1.13
irb.102       up              2   backup   Active      D  3.303 lcl    10.88.1.12
                                                              vip    10.88.1.14
                                                              mas    10.88.1.13
irb.103       up              3   backup   Active      D  2.798 lcl    10.33.1.12
                                                              vip    10.33.1.14
                                                              mas    10.33.1.13
```

## L3 Exit With eBGP Routing Protocol

**NOTE**: When you create any VLAN or VRF creation with campus fabric remember the following best practices:

- Create all VLANs in a switch template and then import them into the **Campus Fabric Dialogue**. Creating the VLANs anywhere else in the Mist GUI ultimately leads to inconsistency which makes it hard to resolve issues.
- If needed, the fabric creates any required VRFs. Do not create VRFs manually elsewhere in the Mist GUI.
- We recommend that you create port profiles within switch templates so that any changes are in sync on all switches in a fabric.

Before you begin, you need a plan for:

- How to implement the routing protocol and route exchange
- How to configure the P2P links
- How to distribute the VLAN assignment that is indirectly used to identify the VRF

Even if the VRF already exists elsewhere in the fabric, such as on the access switch for IP Clos, the system will automatically re-create it on all service block functions when doing an L3 exit.

For each WAN router, you must reuse a VLAN name on a VRF to help the automatic creation of VRFs on the service block function. Keep in mind that when you define the local P2P links and reuse the VLAN, those definitions are purely local, so they do not conflict with the overlay VLAN definition. Additionally, you do not need to define special transport VLANs here. However, you can still use and define special transport VLANs for the P2P links if that better suits your needs.

When defining the P2P links, you must ensure that they are outside of the address range in use by the fabric. The default range used by the fabric for these links is 10.255.240.0/20. We recommend that you a /31 netmask. With that plan, you can use the even number IP addresses for the WAN router side and the odd IP addresses for the fabric side.

The system requires that you use a VLAN for each P2P link on a physical cable. This allows you to have multiple VRFs multiplexed on a single uplink cable. Remember the VLAN internally refers back to the VRF.

For eBGP you must also define your own private ASN for peering. By hardcoded default, the fabric uses 65000 ASN for the EVPN control plane and starts allocating configurable ASN at 65001. After that, it advances one digit for each node. Therefore, we recommend using ASN values below 65000 to avoid

conflict with system assigned ASN. The QFX switch only allows 16 local ASN. Therefore, we recommend that you use a shared ASN among all VRFs. However, in our example, we decided to use a different ASN per WAN router.

Figure 30 on page 51 shows how the two service block functions of the fabric would connect to the first WAN router.

**Figure 30: L3 eBGP-Based Fabric to WAN Router1 Attach**

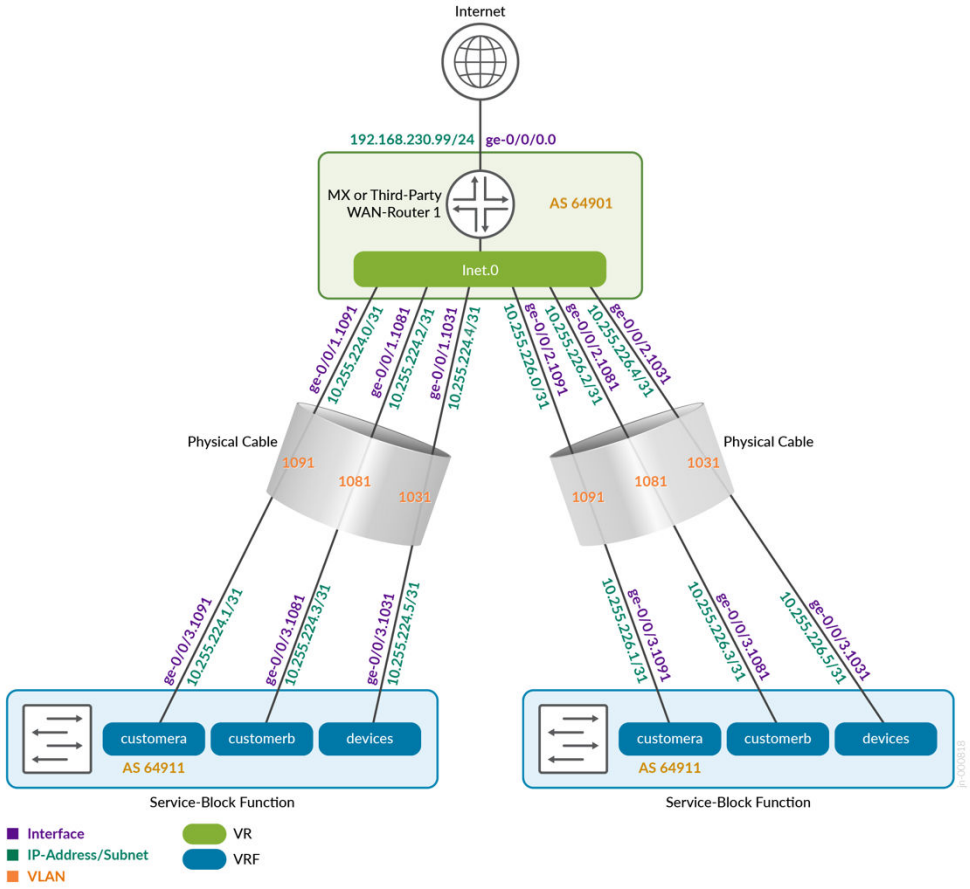



Figure 31 on page 52 shows how the two service block functions of the fabric would connect to the second WAN router. Notice that we now use the second block of VLANs from each VRF.

**Figure 31: L3 eBGP-Based Fabric to WAN Router2 Attach**

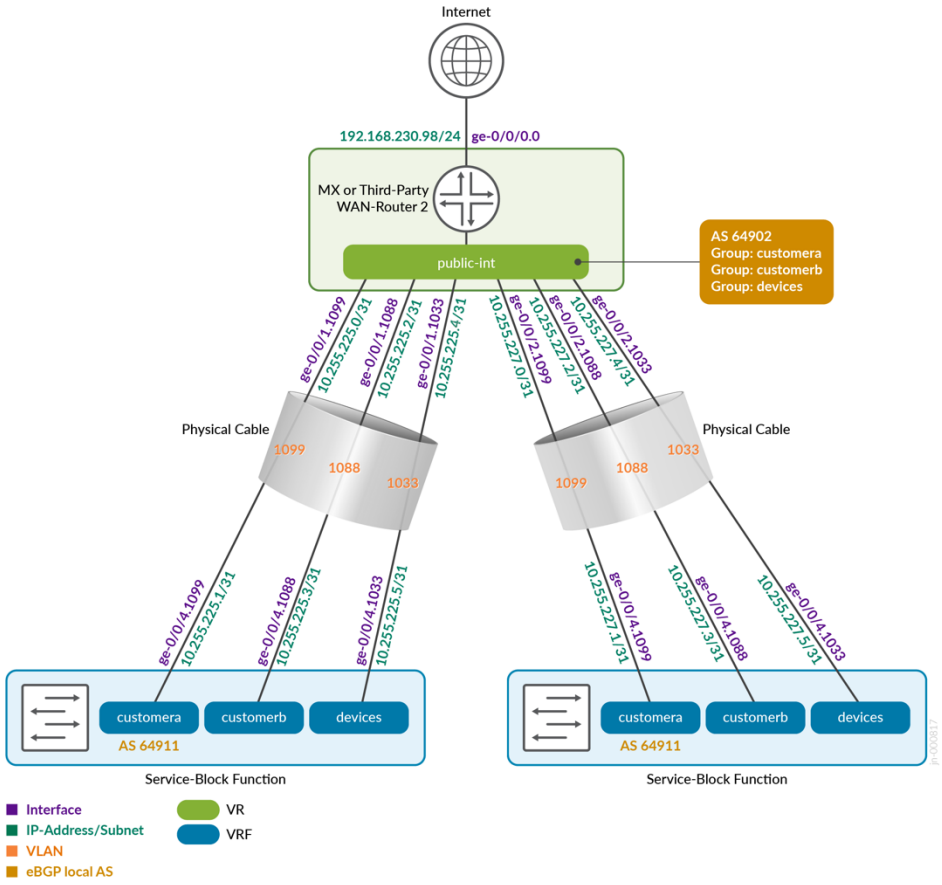



Table 1 on page 52 displays the full configuration between the core1 and core2 switches, as service block function, and the two WAN routers. You can also see the ASN chosen for eBGP.

**Table 1: MX WAN Router and Core Switch Configuration Summary for eBGP Exit**

| Switch | Switch AS | VRF | Core P2P IP | Core IF | WAN Router | WAN Router P2P IP | WAN Router AS | WAN Router IF | VLAN-ID |
|---|---|---|---|---|---|---|---|---|---|
| core1 | 64911 | customera | 10.255.224.1/31 | ge-0/0/3.1091 | **wanrouter1** | 10.255.224.0/31 | 64901 | ge-0/0/1.1091 | 1091 |
| core1 | 64911 | customerb | 10.255.224.3/31 | ge-0/0/3.1081 | **wanrouter1** | 10.255.224.2/31 | 64901 | ge-0/0/1.1081 | 1081 |

**Table 1: MX WAN Router and Core Switch Configuration Summary for eBGP Exit *(Continued)***

| Switch | Switch AS | VRF | Core P2P IP | Core IF | WAN Router | WAN Router P2P IP | WAN Router AS | WAN Router IF | VLAN-ID |
|---|---|---|---|---|---|---|---|---|---|
| core1 | 64911 | devices | 10.255.224.5/31 | ge-0/0/3.1031 | **wanrouter1** | 10.255.224.4/31 | 64901 | ge-0/0/1.1031 | 1031 |
| core1 | 64911 | customera | 10.255.225.1/31 | ge-0/0/4.1099 | wanrouter2 | 10.255.225.0/31 | 64902 | ge-0/0/1.1099 | 1099 |
| core1 | 64911 | customerb | 10.255.225.3/31 | ge-0/0/4.1088 | wanrouter2 | 10.255.225.2/31 | 64902 | ge-0/0/1.1088 | 1088 |
| core1 | 64911 | devices | 10.255.225.5/31 | ge-0/0/4.1033 | wanrouter2 | 10.255.225.4/31 | 64902 | ge-0/0/1.1033 | 1033 |
| core2 | 64911 | customera | 10.255.226.1/31 | ge-0/0/3.1091 | wanrouter1 | 10.255.226.0/31 | 64901 | ge-0/0/2.1091 | 1091 |
| core2 | 64911 | customerb | 10.255.226.3/31 | ge-0/0/3.1081 | wanrouter1 | 10.255.226.2/31 | 64901 | ge-0/0/2.1081 | 1081 |
| core2 | 64911 | devices | 10.255.226.5/31 | ge-0/0/3.1031 | wanrouter1 | 10.255.226.4/31 | 64901 | ge-0/0/2.1031 | 1031 |
| core2 | 64911 | customera | 10.255.227.1/31 | ge-0/0/4.1099 | wanrouter2 | 10.255.227.0/31 | 64902 | ge-0/0/2.1099 | 1099 |
| core2 | 64911 | customerb | 10.255.227.3/31 | ge-0/0/4.1088 | wanrouter2 | 10.255.227.2/31 | 64902 | ge-0/0/2.1088 | 1088 |
| core2 | 64911 | devices | 10.255.227.5/31 | ge-0/0/4.1033 | wanrouter2 | 10.255.227.4/31 | 64902 | ge-0/0/2.1033 | 1033 |

The code block below shows the exported version of the switch template used in this fabric. This allows you to review our setup when importing. As you can see, we have a minimum of two VLANs per VRF. Remember that the L3 exit model requires one VLAN per WAN router and VRF).


```
{
  "additional_config_cmds": [],
  "networks": {
    "vlan1099": {
      "vlan_id": 1099,
      "subnet": "10.99.99.0/24"
    },
    "vlan1088": {
      "vlan_id": 1088,
      "subnet": "10.88.88.0/24"
    },
    "vlan1033": {
      "vlan_id": 1033,
      "subnet": "10.33.33.0/24"
    },
    "vlan1091": {
      "vlan_id": 1091,
      "subnet": "10.99.91.0/24"
    },
    "vlan1081": {
      "vlan_id": 1081,
      "subnet": "10.88.81.0/24"
    },
    "vlan1031": {
      "vlan_id": 1031,
      "subnet": "10.33.31.0/24"
    }
  },
  "port_usages": {
    "vlan1099": {
      "mode": "access",
      "disabled": false,
      "port_network": "vlan1099",
      "voip_network": null,
      "stp_edge": false,
      "mac_auth_protocol": null,
      "all_networks": false,
      "networks": null,
```

```
      "port_auth": null,
      "enable_mac_auth": null,
      "mac_auth_only": null,
      "guest_network": null,
      "bypass_auth_when_server_down": null,
      "speed": "auto",
      "duplex": "auto",
      "mac_limit": 0,
      "persist_mac": false,
      "poe_disabled": false,
      "enable_qos": false,
      "storm_control": {},
      "mtu": null,
      "description": "",
      "disable_autoneg": false
    },
    "vlan1088": {
      "mode": "access",
      "disabled": false,
      "port_network": "vlan1088",
      "voip_network": null,
      "stp_edge": false,
      "mac_auth_protocol": null,
      "all_networks": false,
      "networks": null,
      "port_auth": null,
      "enable_mac_auth": null,
      "mac_auth_only": null,
      "guest_network": null,
      "bypass_auth_when_server_down": null,
      "speed": "auto",
      "duplex": "auto",
      "mac_limit": 0,
      "persist_mac": false,
      "poe_disabled": false,
      "enable_qos": false,
      "storm_control": {},
      "mtu": null,
      "description": "",
      "disable_autoneg": false
    },
    "dynamic": {
      "mode": "dynamic",
```

```
      "reset_default_when": "link_down",
      "rules": []
    }
  },
  "switch_matching": {
    "enable": true,
    "rules": []
  },
  "switch_mgmt": {
    "config_revert_timer": 10,
    "root_password": "<password>",
    "protect_re": {
      "enabled": false
    },
    "tacacs": {
      "enabled": false
    }
  },
  "mist_nac": {
    "enabled": true,
    "network": null
  },
  "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
  },
  "dns_servers": [],
  "dns_suffix": [],
  "ntp_servers": [],
  "acl_policies": [],
  "port_mirroring": {},
  "name": "campus-fabric"
}
```


Within the **Campus Fabric Configuration** dialogue, there is a page called **Configure Networks**. This is where you import your six VLAN's from the switch template. The resulting configuration is shown in the following figures.

**Figure 32: Network and Other IP Configuration**

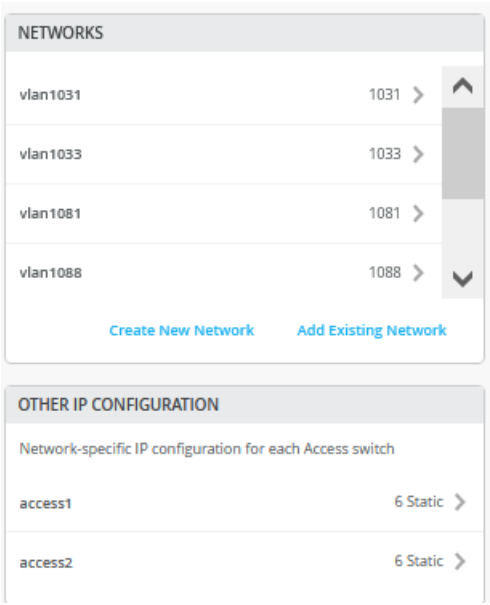
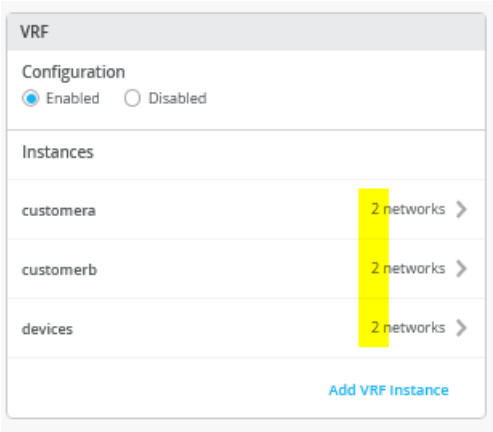


**Figure 33: Create 3 VRF Instances and Attach 2 Networks to Each**




Then you go to each VRF and delete all manual routes you may have. Make sure each VRF has a minimum of two VLAN's attached as those are used to identify the VRF later.

**Figure 34: VRF1 Configure 2 VLANs and Remove All Extra Routes**

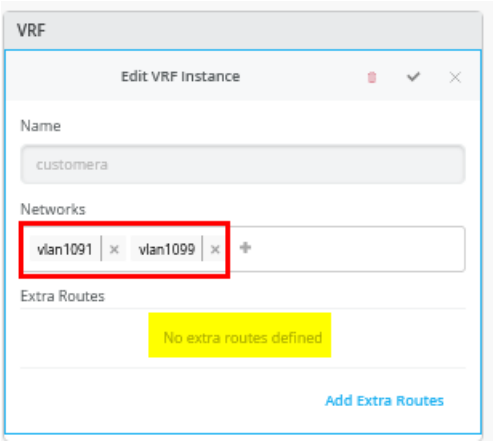


**Figure 35: VRF2 Configure 2 VLANs and Remove All Extra Routes**

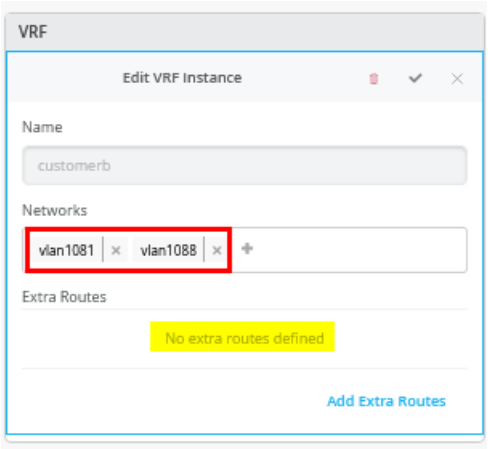
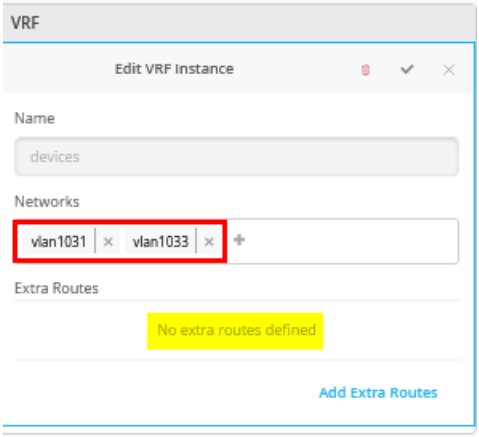



**Figure 36: VRF3 Configure 2 VLANs and Remove All Extra Routes**




### Core1 Switch Configuration

In this example, the service block function is virtual and co-located on the core switch. Therefore, you must configure the two core switches. The following block of pseudocode describes what you need to configure on the core1 switch:

```
# configure the Additional IP-Subnet 10.255.224.1 255.255.255.254 to Network/VLAN:vlan1091
# configure the Additional IP-Subnet 10.255.224.3 255.255.255.254 to Network/VLAN:vlan1081
# configure the Additional IP-Subnet 10.255.224.5 255.255.255.254 to Network/VLAN:vlan1031
# Then bind these 3 Network/VLANs to Port Interface ge-0/0/3 as L3-Sub-Interfaces with MTU=9018
#
# configure the Additional IP-Subnet 10.255.225.1 255.255.255.254 to Network/VLAN:vlan1099
# configure the Additional IP-Subnet 10.255.225.3 255.255.255.254 to Network/VLAN:vlan1088
```

```
# configure the Additional IP-Subnet 10.255.225.5 255.255.255.254 to Network/VLAN:vlan1033
# Then bind these 3 Network/VLANs to Port Interface ge-0/0/4 as L3-Sub-Interfaces with MTU=9018
#
# Enable BGP
# Create an Export policy called 'export-vrfs'
# Add to this export Policy the following Networks as:
# - Add Term w. Name=fabric-all-no-hosts  Prefix=0.0.0.0/0-30  Protocol=None  Then=Accept
# - Add Term w. Name=overlaylo0 Prefix=172.16.192.0/24-32  Protocol=None  Then=Accept
#
# Create an Export policy called 'import-default'
# - Name=default  Prefix=0.0.0.0/0  Protocol=BGP  Action=Accept
#
# Create a BGP Group with:
# - Name=customera0
# - Type=External
# - Network (VLAN)=vlan1091
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.224.0  Neighbor_AS=64901  Hold-Time=90
#
# Create a BGP Group with:
# - Name=customerb0
# - Type=External
# - Network (VLAN)=vlan1081
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.224.2  Neighbor_AS=64901  Hold-Time=90
#
# Create a BGP Group with:
# - Name=devices0
# - Type=External
# - Network (VLAN)=vlan1031
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs and Import=import-default
# Add also the following Neighbor
```

```
# - IP_Address=10.255.224.4  Neighbor_AS=64901  Hold-Time=90
#
# Create a BGP Group with:
# - Name=customera1
# - Type=External
# - Network (VLAN)=vlan1099
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.225.0  Neighbor_AS=64902  Hold-Time=90
#
# Create a BGP Group with:
# - Name=customerb1
# - Type=External
# - Network (VLAN)=vlan1088
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.225.2  Neighbor_AS=64902  Hold-Time=90
#
# Create a BGP Group with:
# - Name=devices1
# - Type=External
# - Network (VLAN)=vlan1033
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.225.4  Neighbor_AS=64902  Hold-Time=90
```

The following screenshots show the previous configuration translated into the Mist GUI. We start with the additional IP configuration. Notice that the VLAN IP addresses do not match the IP addresses that these VLANs had originally in the overlay. This is by design. You must always have the VLAN as a reference back to the VRF.

**Figure 37: Additional IP Configuration**

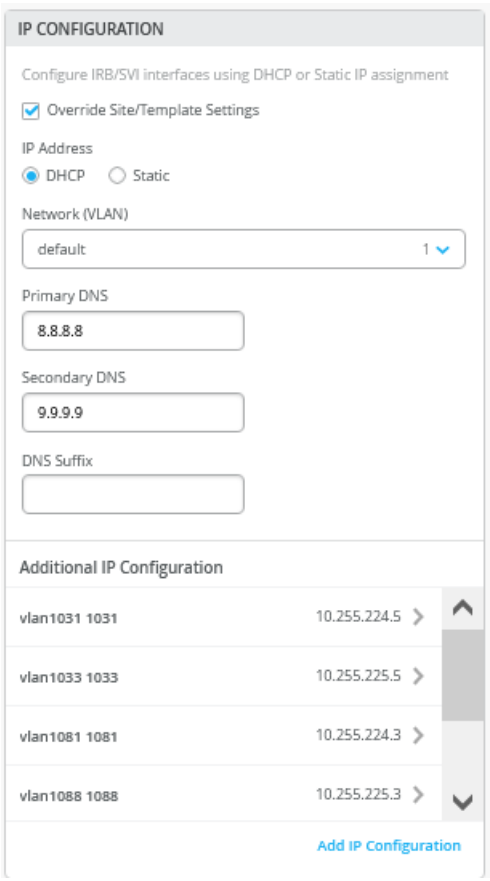
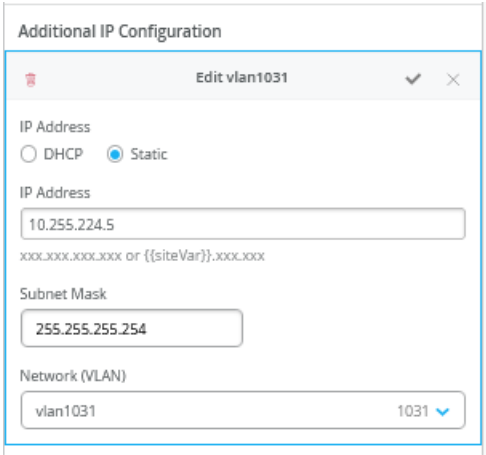
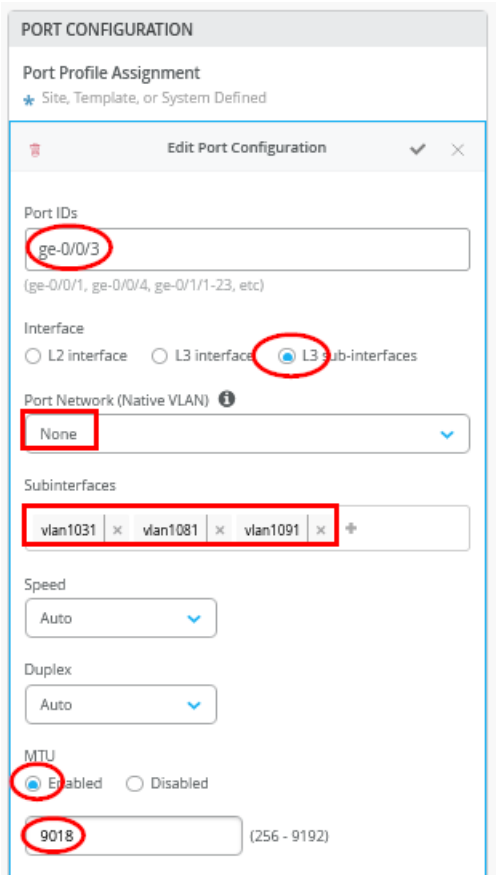


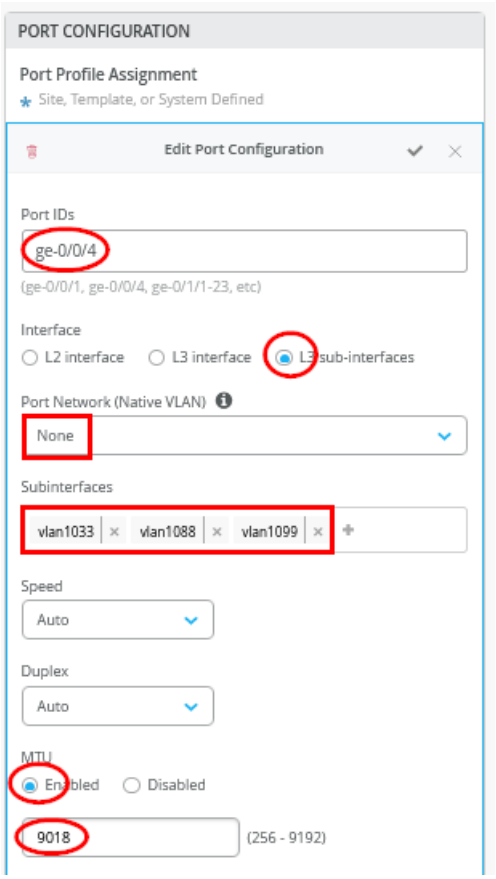

**Figure 38: One of Six VLANs to Configure**




In the **Port Configuration** window, you must enable the **L3-sub-interfaces** and assign the first 3 sub-interfaces defined.

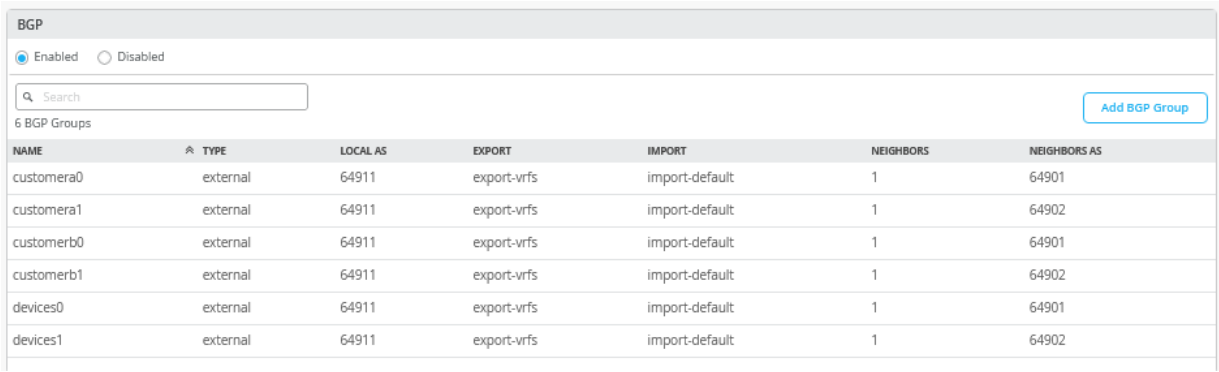



In the second **Port Configuration** window, towards the other WAN router, assign the second 3 sub-interfaces defined.




Next, you must enter the entire eBGP configuration with all six peers (three VRFs and two WAN routers. When finished, the overview page should be as shown in Figure 39 on page 64 .

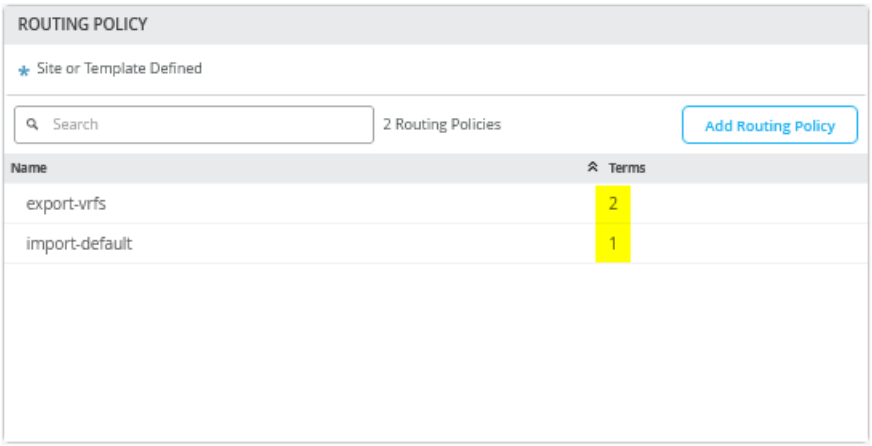
**Figure 39: Complete eBGP Configuration**




| NAME | TYPE | LOCAL AS | EXPORT | IMPORT | NEIGHBORS | NEIGHBORS AS |
|---|---|---|---|---|---|---|
| customera0 | external | 64911 | export-vrfs | import-default | 1 | 64901 |
| customera1 | external | 64911 | export-vrfs | import-default | 1 | 64902 |
| customerb0 | external | 64911 | export-vrfs | import-default | 1 | 64901 |
| customerb1 | external | 64911 | export-vrfs | import-default | 1 | 64902 |
| devices0 | external | 64911 | export-vrfs | import-default | 1 | 64901 |
| devices1 | external | 64911 | export-vrfs | import-default | 1 | 64902 |

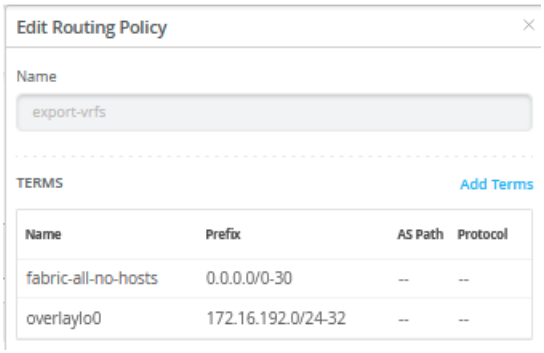
First, you define two routing policies, a summary of which is shown in the above table.

**Figure 40: Routing Policy Summary**




In our example, the export route policies contain the following terms:

- Term=fabric-all-no-hosts — Used to automatically export all routes of a fabric from all configured VRFs and their VLANs. We strip potential P2P links and single host routes. To achieve this, we use the prefix “0.0.0.0/0-30”. Using “0-30” in the prefix is a way of defining “orlonger” in the Junos OS.
- Term=overlaylo0 — Used to export the overlay loopback IP addresses (for DHCP relay usage) down to the host IP level as they are usually individually assigned to lo0.x. To achieve this, we use 172.16.192.0/24-32 as we here need to go down towards host level which we do not do in the above statement.




If desired, you can simplify the rule set by using a single term such as “0.0.0.0/0-32,” which exports all routes—including those for individual clients in the fabric, represented by their host IP addresses. This approach can also be useful for debugging, as examining the AS-Path information on the WAN router will reveal which EVPN fabric switch is announcing a given host IP.

The downside, however, is that this approach may result in a large number of routes being exchanged with the WAN router, whereas the default setting of “0.0.0.0/0-30” limits route sharing to prefixes associated only with the configured VLANs.

```
10.99.99.0/24      *[BGP/170] 00:06:46, localpref 100, from 10.255.226.1
                      AS path: 64911 65001 65004 65005 I, validation-state: unverified
                       to 10.255.224.1 via ge-0/0/2.1091
                       to 10.255.226.1 via ge-0/0/3.1091
                    >  to 10.255.225.1 via ge-7/0/2.1099
                       to 10.255.227.1 via ge-7/0/3.1099
                    [BGP/170] 00:06:50, localpref 100
                      AS path: 64911 65002 65004 65005 I, validation-state: unverified
                    >  to 10.255.224.1 via ge-0/0/2.1091
                    [BGP/170] 00:12:38, localpref 100
                      AS path: 64911 65002 65004 65005 I, validation-state: unverified
                    >  to 10.255.225.1 via ge-7/0/2.1099
                    [BGP/170] 00:06:46, localpref 100
                      AS path: 64911 65001 65004 65005 I, validation-state: unverified
                    >  to 10.255.227.1 via ge-7/0/3.1099
10.99.99.99/32     *[BGP/170] 00:00:14, localpref 100, from 10.255.224.1
                      AS path: 64911 65002 65003 65006 I, validation-state: unverified
                       to 10.255.224.1 via ge-0/0/2.1091
                       to 10.255.226.1 via ge-0/0/3.1091
                    >  to 10.255.225.1 via ge-7/0/2.1099
                       to 10.255.227.1 via ge-7/0/3.1099
                    [BGP/170] 00:01:33, localpref 100
                      AS path: 64911 65002 65003 65006 I, validation-state: unverified
                    >  to 10.255.225.1 via ge-7/0/2.1099
                    [BGP/170] 00:00:14, localpref 100
                      AS path: 64911 65001 65003 65006 I, validation-state: unverified
                    >  to 10.255.226.1 via ge-0/0/3.1091
                    [BGP/170] 00:00:14, localpref 100
                      AS path: 64911 65001 65003 65006 I, validation-state: unverified
                    >  to 10.255.227.1 via ge-7/0/3.1099
```

The import policy imports the default route from the WAN router.

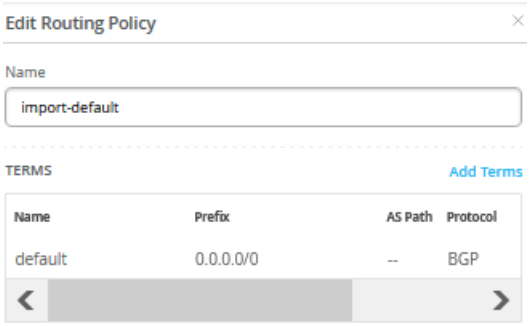
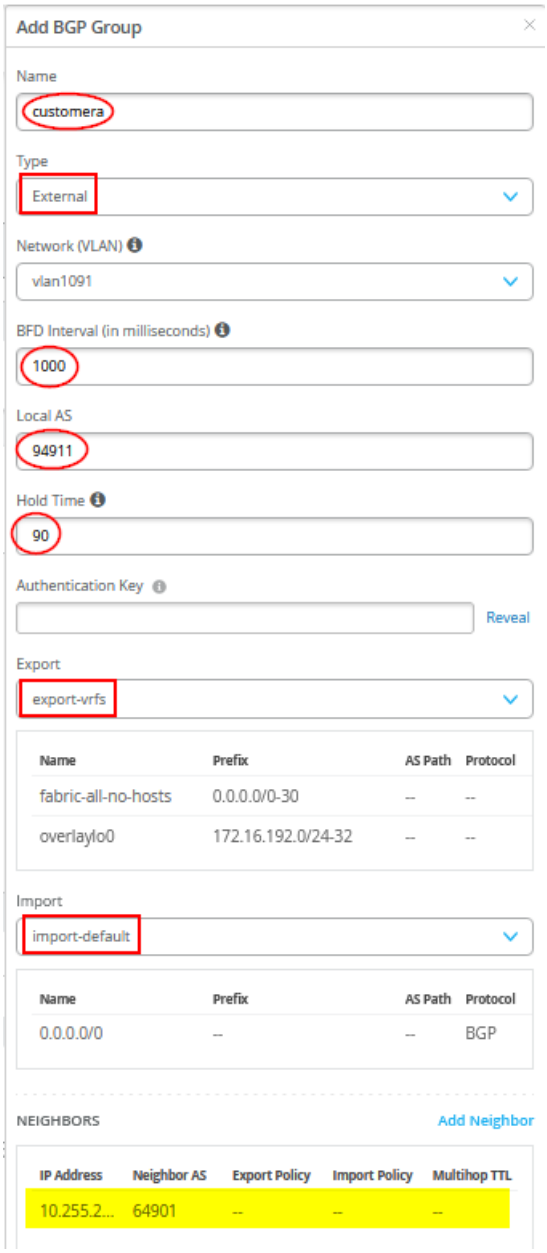
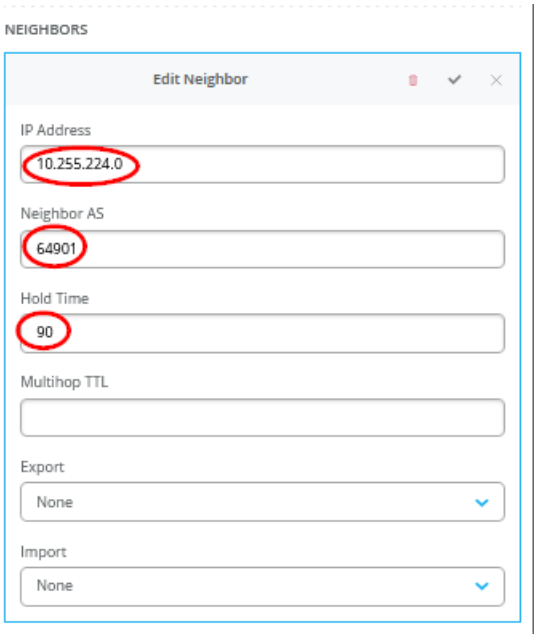



Figure 41 on page 68 shows the configuration of a single BGP peering entry with the required entries called out.

**Figure 41: Example BGP Peering Entry**




You must also define the WAN router as a BGP neighbor.




You may see a warning message as shown in Figure 42 on page 69 . It is safe to ignore those.

**Figure 42: Potential Warning Message**


- The messages mentioned above appear because we are reusing overlay VLANs—already assigned to VRFs—for underlay BGP peering with the WAN router. This can be avoided by using dedicated VLANs for BGP peering that are not part of the fabric's VRF configuration. To do so, a local VRF setup similar to the “L2 Exit with Transport VLAN” approach is required. This involves manually adding the BGP peering VLANs to each VRF, along with the overlay VLANs configured through the standard fabric configuration dialogue.
- While this method eliminates the false-positive error messages, it comes with a trade-off: every time a new VLAN is added to the fabric, you'll need to manually update each service block to include the new VLAN in the custom local VRF configuration.
- In contrast, the method used in this JVD-E avoids such error-prone reconfiguration. You simply add the new VLAN to the switch template and configure it through the fabric dialogue. The service block configuration remains unchanged and will automatically export the new VLAN's routes based on the export filter configuration shown above.

### Core2 Switch Configuration

The following pseudocode represents the configuration you must apply to the core2 switch:

```
# configure the Additional IP-Subnet 10.255.226.1 255.255.255.254 to Network/VLAN:vlan1091
# configure the Additional IP-Subnet 10.255.226.3 255.255.255.254 to Network/VLAN:vlan1081
# configure the Additional IP-Subnet 10.255.226.5 255.255.255.254 to Network/VLAN:vlan1031
# Then bind these 3 Network/VLANs to Port Interface ge-0/0/3 as L3-Sub-Interfaces with MTU=9018
#
# configure the Additional IP-Subnet 10.255.227.1 255.255.255.254 to Network/VLAN:vlan1099
# configure the Additional IP-Subnet 10.255.227.3 255.255.255.254 to Network/VLAN:vlan1088
# configure the Additional IP-Subnet 10.255.227.5 255.255.255.254 to Network/VLAN:vlan1033
# Then bind these 3 Network/VLANs to Port Interface ge-0/0/4 as L3-Sub-Interfaces with MTU=9018
#
# Enable BGP
# Create an Export policy called 'export-vrfs'
# Add to this export Policy the following Networks as:
# - Add Term w. Name=fabric-all-no-hosts  Prefix=0.0.0.0/0-30  Protocol=None  Then=Accept
# - Add Term w. Name=overlaylo0 Prefix=172.16.192.0/24-32  Protocol=None  Then=Accept
#
# Create an Export policy called 'import-default'
# - Name=default  Prefix=0.0.0.0/0  Protocol=BGP  Action=Accept
#
# Create a BGP Group with:
# - Name=customera0
# - Type=External
# - Network (VLAN)=vlan1091
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.226.0  Neighbor_AS=64901  Hold-Time=90
#
# Create a BGP Group with:
# - Name=customerb0
# - Type=External
# - Network (VLAN)=vlan1081
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.226.2  Neighbor_AS=64901  Hold-Time=90
```

```
#
# Create a BGP Group with:
# - Name=devices0
# - Type=External
# - Network (VLAN)=vlan1031
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.226.4  Neighbor_AS=64901  Hold-Time=90
#
# Create a BGP Group with:
# - Name=customera1
# - Type=External
# - Network (VLAN)=vlan1099
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.227.0  Neighbor_AS=64902  Hold-Time=90
#
# Create a BGP Group with:
# - Name=customerb1
# - Type=External
# - Network (VLAN)=vlan1088
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.227.2  Neighbor_AS=64902  Hold-Time=90
#
# Create a BGP Group with:
# - Name=devices1
# - Type=External
# - Network (VLAN)=vlan1033
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs and Import=import-default
```

```
# Add also the following Neighbor
# - IP_Address=10.255.227.4  Neighbor_AS=64902  Hold-Time=90
```

Aside from the P2P subnets and the BGP neighbours, the configuration on the core2 switch is the same as the configuration on the core1 switch.

**Figure 43: Core2 Switch Additional IP Configuration**

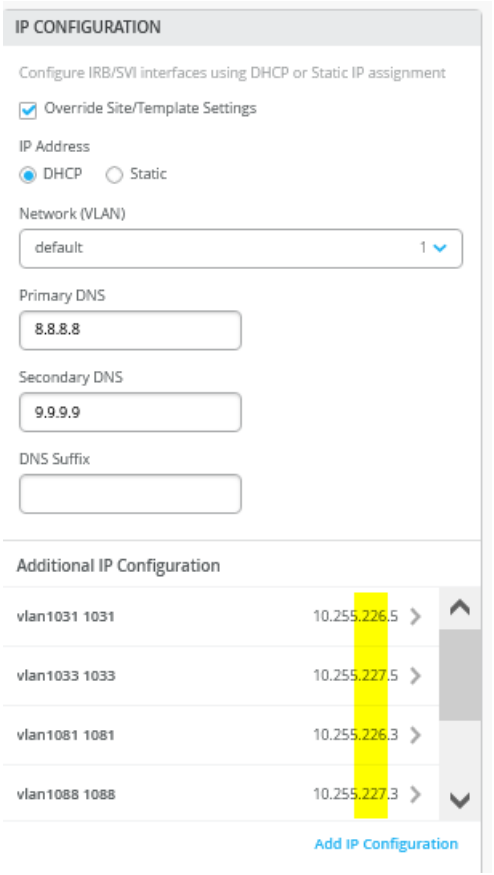
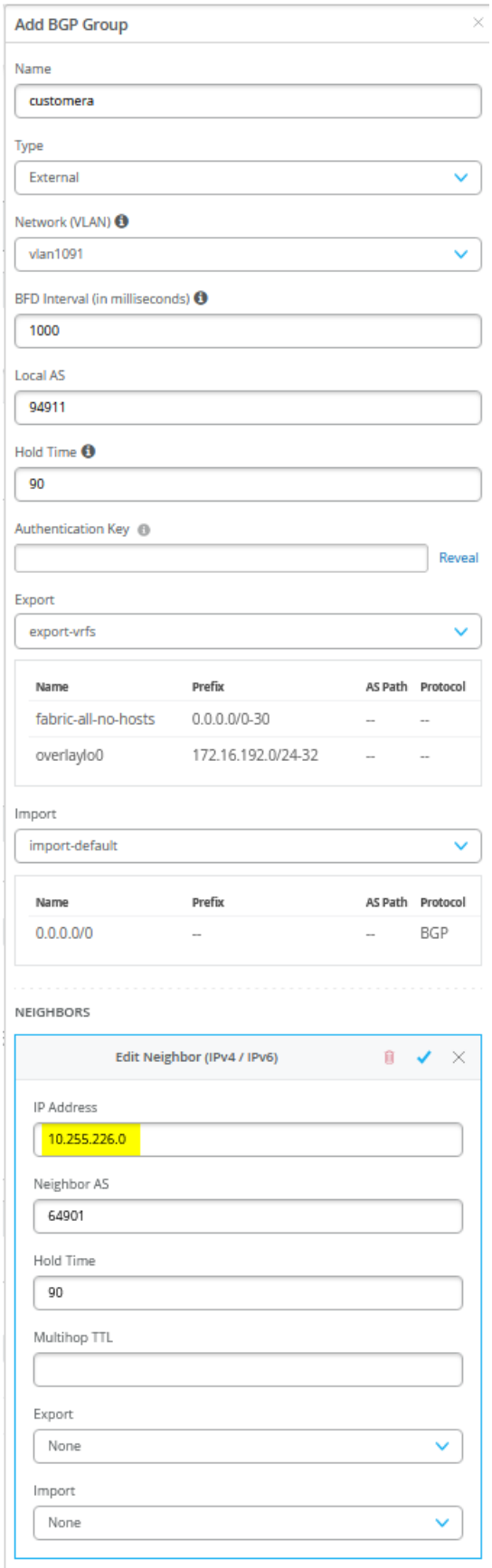


**Figure 44: Core2 Switch BGP Neighbor Configuration**



## Juniper MX as a WAN Router

The following several code blocks show the Junos OS CLI configuration of the P2P interfaces, the entire eBGP config with all BGP neighbours, and all import and export route policies for each WAN router. You may need to add default routes and interfaces to complete the configuration because those need to be signalled to the fabric but we don't know how your device gets to know those.

CLI configuration for the first WAN router:

```
set system host-name wanrouter1
#
delete interfaces ge-0/0/1
set interfaces ge-0/0/1 flexible-vlan-tagging
set interfaces ge-0/0/1 mtu 9014
set interfaces ge-0/0/1 unit 1091 description vlan1091
set interfaces ge-0/0/1 unit 1091 vlan-id 1091
set interfaces ge-0/0/1 unit 1091 family inet address 10.255.224.0/31
set interfaces ge-0/0/1 unit 1081 description vlan1081
set interfaces ge-0/0/1 unit 1081 vlan-id 1081
set interfaces ge-0/0/1 unit 1081 family inet address 10.255.224.2/31
set interfaces ge-0/0/1 unit 1031 description vlan1031
set interfaces ge-0/0/1 unit 1031 vlan-id 1031
set interfaces ge-0/0/1 unit 1031 family inet address 10.255.224.4/31
#
delete interfaces ge-0/0/2
set interfaces ge-0/0/2 flexible-vlan-tagging
set interfaces ge-0/0/2 mtu 9014
set interfaces ge-0/0/2 unit 1091 description vlan1091
set interfaces ge-0/0/2 unit 1091 vlan-id 1091
set interfaces ge-0/0/2 unit 1091 family inet address 10.255.226.0/31
set interfaces ge-0/0/2 unit 1081 description vlan1081
set interfaces ge-0/0/2 unit 1081 vlan-id 1081
set interfaces ge-0/0/2 unit 1081 family inet address 10.255.226.2/31
set interfaces ge-0/0/2 unit 1031 description vlan1031
set interfaces ge-0/0/2 unit 1031 vlan-id 1031
set interfaces ge-0/0/2 unit 1031 family inet address 10.255.226.4/31
#
# needed in and export policy
delete policy-options
set policy-options policy-statement fabric term 1 from protocol bgp
set policy-options policy-statement fabric term 1 from route-filter 0.0.0.0/0 orlonger
set policy-options policy-statement fabric term 1 then accept
```

```
set policy-options policy-statement fabric term 2 then reject
set policy-options policy-statement internet term 1 from protocol static
set policy-options policy-statement internet term 1 from route-filter 0.0.0.0/0 exact
set policy-options policy-statement internet term 1 then accept
set policy-options policy-statement internet term 2 then reject
#
delete routing-instances public-int
set routing-instances public-int instance-type virtual-router
set routing-instances public-int interface ge-0/0/1.1091
set routing-instances public-int interface ge-0/0/1.1081
set routing-instances public-int interface ge-0/0/1.1031
set routing-instances public-int interface ge-0/0/2.1091
set routing-instances public-int interface ge-0/0/2.1081
set routing-instances public-int interface ge-0/0/2.1031
#
delete routing-instances public-int protocols bgp group customera
set routing-instances public-int protocols bgp group customera type external
set routing-instances public-int protocols bgp group customera family inet unicast
set routing-instances public-int protocols bgp group customera multipath multiple-as
set routing-instances public-int protocols bgp group customera local-as 64901
set routing-instances public-int protocols bgp group customera hold-time 90
set routing-instances public-int protocols bgp group customera import fabric
set routing-instances public-int protocols bgp group customera export internet
set routing-instances public-int protocols bgp group customera bfd-liveness-detection minimum-
interval 1000
set routing-instances public-int protocols bgp group customera bfd-liveness-detection multiplier
3
set routing-instances public-int protocols bgp group customera bfd-liveness-detection session-
mode automatic
set routing-instances public-int protocols bgp group customera neighbor 10.255.224.1 peer-as
64911
set routing-instances public-int protocols bgp group customera neighbor 10.255.226.1 peer-as
64911
#
delete routing-instances public-int protocols bgp group customerb
set routing-instances public-int protocols bgp group customerb type external
set routing-instances public-int protocols bgp group customerb family inet unicast
set routing-instances public-int protocols bgp group customerb multipath multiple-as
set routing-instances public-int protocols bgp group customerb local-as 64901
set routing-instances public-int protocols bgp group customerb hold-time 90
set routing-instances public-int protocols bgp group customerb import fabric
set routing-instances public-int protocols bgp group customerb export internet
set routing-instances public-int protocols bgp group customerb bfd-liveness-detection minimum-
```

```
interval 1000
set routing-instances public-int protocols bgp group customerb bfd-liveness-detection multiplier
3
set routing-instances public-int protocols bgp group customerb bfd-liveness-detection session-
mode automatic
set routing-instances public-int protocols bgp group customerb neighbor 10.255.224.3 peer-as
64911
set routing-instances public-int protocols bgp group customerb neighbor 10.255.226.3 peer-as
64911
#
delete routing-instances public-int protocols bgp group devices
set routing-instances public-int protocols bgp group devices type external
set routing-instances public-int protocols bgp group devices family inet unicast
set routing-instances public-int protocols bgp group devices multipath multiple-as
set routing-instances public-int protocols bgp group devices local-as 64901
set routing-instances public-int protocols bgp group devices hold-time 90
set routing-instances public-int protocols bgp group devices import fabric
set routing-instances public-int protocols bgp group devices export internet
set routing-instances public-int protocols bgp group devices bfd-liveness-detection minimum-
interval 1000
set routing-instances public-int protocols bgp group devices bfd-liveness-detection multiplier 3
set routing-instances public-int protocols bgp group devices bfd-liveness-detection session-mode
automatic
set routing-instances public-int protocols bgp group devices neighbor 10.255.224.5 peer-as 64911
set routing-instances public-int protocols bgp group devices neighbor 10.255.226.5 peer-as 64911
```

Configuration for the second WAN router:

```
set system host-name wanrouter2
#
delete interfaces ge-0/0/1
set interfaces ge-0/0/1 flexible-vlan-tagging
set interfaces ge-0/0/1 mtu 9014
set interfaces ge-0/0/1 unit 1099 description vlan1099
set interfaces ge-0/0/1 unit 1099 vlan-id 1099
set interfaces ge-0/0/1 unit 1099 family inet address 10.255.225.0/31
set interfaces ge-0/0/1 unit 1088 description vlan1088
set interfaces ge-0/0/1 unit 1088 vlan-id 1088
set interfaces ge-0/0/1 unit 1088 family inet address 10.255.225.2/31
set interfaces ge-0/0/1 unit 1033 description vlan1033
set interfaces ge-0/0/1 unit 1033 vlan-id 1033
set interfaces ge-0/0/1 unit 1033 family inet address 10.255.225.4/31
```

```
#
delete interfaces ge-0/0/2
set interfaces ge-0/0/2 flexible-vlan-tagging
set interfaces ge-0/0/2 mtu 9014
set interfaces ge-0/0/2 unit 1099 description vlan1099
set interfaces ge-0/0/2 unit 1099 vlan-id 1099
set interfaces ge-0/0/2 unit 1099 family inet address 10.255.227.0/31
set interfaces ge-0/0/2 unit 1088 description vlan1088
set interfaces ge-0/0/2 unit 1088 vlan-id 1088
set interfaces ge-0/0/2 unit 1088 family inet address 10.255.227.2/31
set interfaces ge-0/0/2 unit 1033 description vlan1033
set interfaces ge-0/0/2 unit 1033 vlan-id 1033
set interfaces ge-0/0/2 unit 1033 family inet address 10.255.227.4/31
#
# needed in and export policy
delete policy-options
set policy-options policy-statement fabric term 1 from protocol bgp
set policy-options policy-statement fabric term 1 from route-filter 0.0.0.0/0 orlonger
set policy-options policy-statement fabric term 1 then accept
set policy-options policy-statement fabric term 2 then reject
set policy-options policy-statement internet term 1 from protocol static
set policy-options policy-statement internet term 1 from route-filter 0.0.0.0/0 exact
set policy-options policy-statement internet term 1 then accept
set policy-options policy-statement internet term 2 then reject
#
delete routing-instances public-int
set routing-instances public-int instance-type virtual-router
set routing-instances public-int interface ge-0/0/1.1099
set routing-instances public-int interface ge-0/0/1.1088
set routing-instances public-int interface ge-0/0/1.1033
set routing-instances public-int interface ge-0/0/2.1099
set routing-instances public-int interface ge-0/0/2.1088
set routing-instances public-int interface ge-0/0/2.1033
#
delete routing-instances public-int protocols bgp group customera
set routing-instances public-int protocols bgp group customera type external
set routing-instances public-int protocols bgp group customera family inet unicast
set routing-instances public-int protocols bgp group customera multipath multiple-as
set routing-instances public-int protocols bgp group customera local-as 64902
set routing-instances public-int protocols bgp group customera hold-time 90
set routing-instances public-int protocols bgp group customera import fabric
set routing-instances public-int protocols bgp group customera export internet
set routing-instances public-int protocols bgp group customera bfd-liveness-detection minimum-
```

```
interval 1000
set routing-instances public-int protocols bgp group customera bfd-liveness-detection multiplier
3
set routing-instances public-int protocols bgp group customera bfd-liveness-detection session-
mode automatic
set routing-instances public-int protocols bgp group customera neighbor 10.255.225.1 peer-as
64911
set routing-instances public-int protocols bgp group customera neighbor 10.255.227.1 peer-as
64911
#
delete routing-instances public-int protocols bgp group customerb
set routing-instances public-int protocols bgp group customerb type external
set routing-instances public-int protocols bgp group customerb family inet unicast
set routing-instances public-int protocols bgp group customerb multipath multiple-as
set routing-instances public-int protocols bgp group customerb local-as 64902
set routing-instances public-int protocols bgp group customerb hold-time 90
set routing-instances public-int protocols bgp group customerb import fabric
set routing-instances public-int protocols bgp group customerb export internet
set routing-instances public-int protocols bgp group customerb bfd-liveness-detection minimum-
interval 1000
set routing-instances public-int protocols bgp group customerb bfd-liveness-detection multiplier
3
set routing-instances public-int protocols bgp group customerb bfd-liveness-detection session-
mode automatic
set routing-instances public-int protocols bgp group customerb neighbor 10.255.225.3 peer-as
64911
set routing-instances public-int protocols bgp group customerb neighbor 10.255.227.3 peer-as
64911
#
delete routing-instances public-int protocols bgp group devices
set routing-instances public-int protocols bgp group devices type external
set routing-instances public-int protocols bgp group devices family inet unicast
set routing-instances public-int protocols bgp group devices multipath multiple-as
set routing-instances public-int protocols bgp group devices local-as 64902
set routing-instances public-int protocols bgp group devices hold-time 90
set routing-instances public-int protocols bgp group devices import fabric
set routing-instances public-int protocols bgp group devices export internet
set routing-instances public-int protocols bgp group devices bfd-liveness-detection minimum-
interval 1000
set routing-instances public-int protocols bgp group devices bfd-liveness-detection multiplier 3
set routing-instances public-int protocols bgp group devices bfd-liveness-detection session-mode
automatic
```

```
set routing-instances public-int protocols bgp group devices neighbor 10.255.225.5 peer-as 64911
set routing-instances public-int protocols bgp group devices neighbor 10.255.227.5 peer-as 64911
```

You can use the following CLI commands to assist with debugging on WAN router1.

```
root@wanrouter1> 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         30         27       0       0       11:37 Establ
  public-int.inet.0: 2/2/2/0
10.255.224.3          64911         29         27       0       0       11:31 Establ
  public-int.inet.0: 2/2/2/0
10.255.224.5          64911         29         27       0       0       11:37 Establ
  public-int.inet.0: 1/1/1/0
10.255.226.1          64911         30         27       0       0       11:30 Establ
  public-int.inet.0: 2/2/2/0
10.255.226.3          64911         30         27       0       0       11:39 Establ
  public-int.inet.0: 2/2/2/0
10.255.226.5          64911         29         27       0       0       11:38 Establ
  public-int.inet.0: 1/1/1/0
.
root@wanrouter1> show route table public-int.inet.0
.
public-int.inet.0: 23 destinations, 25 routes (23 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
0.0.0.0/0          *[Static/5] 00:00:42
                    >  to 192.168.230.1 via ge-0/0/0.0
10.88.88.0/24      *[BGP/170] 00:11:47, localpref 100
                      AS path: 64911 65002 65003 65005 I, validation-state: unverified
                       to 10.255.224.3 via ge-0/0/1.1081
                    >  to 10.255.226.3 via ge-0/0/2.1081
                    [BGP/170] 00:11:47, localpref 100
                      AS path: 64911 65001 65003 65005 I, validation-state: unverified
                    >  to 10.255.224.3 via ge-0/0/1.1081
10.99.99.0/24      *[BGP/170] 00:11:46, localpref 100, from 10.255.224.1
                      AS path: 64911 65001 65003 65005 I, validation-state: unverified
                       to 10.255.224.1 via ge-0/0/1.1091
                    >  to 10.255.226.1 via ge-0/0/2.1091
                    [BGP/170] 00:11:46, localpref 100
```

```
                    AS path: 64911 65002 65003 65005 I, validation-state: unverified
                  >  to 10.255.226.1 via ge-0/0/2.1091
10.255.224.0/31    *[Direct/0] 00:22:31
                  >  via ge-0/0/1.1091
10.255.224.0/32    *[Local/0] 00:22:31
                     Local via ge-0/0/1.1091
10.255.224.2/31    *[Direct/0] 00:22:31
                  >  via ge-0/0/1.1081
10.255.224.2/32    *[Local/0] 00:22:31
                     Local via ge-0/0/1.1081
10.255.224.4/31    *[Direct/0] 00:22:31
                  >  via ge-0/0/1.1031
10.255.224.4/32    *[Local/0] 00:22:31
                     Local via ge-0/0/1.1031
10.255.226.0/31    *[Direct/0] 00:22:31
                  >  via ge-0/0/2.1091
10.255.226.0/32    *[Local/0] 00:22:31
                     Local via ge-0/0/2.1091
10.255.226.2/31    *[Direct/0] 00:22:31
                  >  via ge-0/0/2.1081
10.255.226.2/32    *[Local/0] 00:22:31
                     Local via ge-0/0/2.1081
10.255.226.4/31    *[Direct/0] 00:22:31
                  >  via ge-0/0/2.1031
10.255.226.4/32    *[Local/0] 00:22:31
                     Local via ge-0/0/2.1031
172.16.193.1/32    *[BGP/170] 00:11:53, localpref 100
                    AS path: 64911 I, validation-state: unverified
                  >  to 10.255.224.1 via ge-0/0/1.1091
172.16.193.2/32    *[BGP/170] 00:11:46, localpref 100
                    AS path: 64911 I, validation-state: unverified
                  >  to 10.255.226.1 via ge-0/0/2.1091
172.16.194.1/32    *[BGP/170] 00:11:47, localpref 100
                    AS path: 64911 I, validation-state: unverified
                  >  to 10.255.224.3 via ge-0/0/1.1081
172.16.194.2/32    *[BGP/170] 00:11:55, localpref 100
                    AS path: 64911 I, validation-state: unverified
                  >  to 10.255.226.3 via ge-0/0/2.1081
172.16.195.1/32    *[BGP/170] 00:11:53, localpref 100
                    AS path: 64911 I, validation-state: unverified
                  >  to 10.255.224.5 via ge-0/0/1.1031
172.16.195.2/32    *[BGP/170] 00:11:54, localpref 100
                    AS path: 64911 I, validation-state: unverified
```

```
                    >  to 10.255.226.5 via ge-0/0/2.1031
192.168.230.0/24   *[Direct/0] 00:01:08
                    >  via ge-0/0/0.0
192.168.230.99/32  *[Local/0] 00:01:08
                       Local via ge-0/0/0.0
```

You can use the following CLI commands to assist with debugging on WAN router2.

```
root@wanrouter2> 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...
10.255.225.1          64911         36         33       0       0       14:29 Establ
  public-int.inet.0: 2/2/2/0
10.255.225.3          64911         36         33       0       0       14:21 Establ
  public-int.inet.0: 2/2/2/0
10.255.225.5          64911         35         34       0       0       14:31 Establ
  public-int.inet.0: 1/1/1/0
10.255.227.1          64911         36         34       0       0       14:35 Establ
  public-int.inet.0: 2/2/2/0
10.255.227.3          64911         36         34       0       0       14:35 Establ
  public-int.inet.0: 2/2/2/0
10.255.227.5          64911         34         33       0       0       14:21 Establ
  public-int.inet.0: 1/1/1/0
.
root@wanrouter2> show route table public-int.inet.0
.
public-int.inet.0: 23 destinations, 25 routes (23 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
0.0.0.0/0          *[Static/5] 00:00:24
                    >  to 192.168.230.1 via ge-0/0/0.0
10.88.88.0/24      *[BGP/170] 00:14:35, localpref 100
                      AS path: 64911 65002 65003 65005 I, validation-state: unverified
                       to 10.255.225.3 via ge-0/0/1.1088
                    >  to 10.255.227.3 via ge-0/0/2.1088
                    [BGP/170] 00:14:35, localpref 100
                      AS path: 64911 65001 65003 65005 I, validation-state: unverified
                    >  to 10.255.225.3 via ge-0/0/1.1088
10.99.99.0/24      *[BGP/170] 00:14:43, localpref 100
                      AS path: 64911 65002 65003 65005 I, validation-state: unverified
```

```
                       to 10.255.225.1 via ge-0/0/1.1099
                     >  to 10.255.227.1 via ge-0/0/2.1099
                     [BGP/170] 00:14:43, localpref 100
                       AS path: 64911 65001 65003 65005 I, validation-state: unverified
                     >  to 10.255.225.1 via ge-0/0/1.1099
10.255.225.0/31    *[Direct/0] 00:25:19
                     >  via ge-0/0/1.1099
10.255.225.0/32    *[Local/0] 00:25:19
                       Local via ge-0/0/1.1099
10.255.225.2/31    *[Direct/0] 00:25:19
                     >  via ge-0/0/1.1088
10.255.225.2/32    *[Local/0] 00:25:19
                       Local via ge-0/0/1.1088
10.255.225.4/31    *[Direct/0] 00:25:19
                     >  via ge-0/0/1.1033
10.255.225.4/32    *[Local/0] 00:25:19
                       Local via ge-0/0/1.1033
10.255.227.0/31    *[Direct/0] 00:25:19
                     >  via ge-0/0/2.1099
10.255.227.0/32    *[Local/0] 00:25:19
                       Local via ge-0/0/2.1099
10.255.227.2/31    *[Direct/0] 00:25:19
                     >  via ge-0/0/2.1088
10.255.227.2/32    *[Local/0] 00:25:19
                       Local via ge-0/0/2.1088
10.255.227.4/31    *[Direct/0] 00:25:19
                     >  via ge-0/0/2.1033
10.255.227.4/32    *[Local/0] 00:25:19
                       Local via ge-0/0/2.1033
172.16.193.1/32    *[BGP/170] 00:14:43, localpref 100
                       AS path: 64911 I, validation-state: unverified
                     >  to 10.255.225.1 via ge-0/0/1.1099
172.16.193.2/32    *[BGP/170] 00:14:49, localpref 100
                       AS path: 64911 I, validation-state: unverified
                     >  to 10.255.227.1 via ge-0/0/2.1099
172.16.194.1/32    *[BGP/170] 00:14:35, localpref 100
                       AS path: 64911 I, validation-state: unverified
                     >  to 10.255.225.3 via ge-0/0/1.1088
172.16.194.2/32    *[BGP/170] 00:14:49, localpref 100
                       AS path: 64911 I, validation-state: unverified
                     >  to 10.255.227.3 via ge-0/0/2.1088
172.16.195.1/32    *[BGP/170] 00:14:45, localpref 100
                       AS path: 64911 I, validation-state: unverified
```

```
                    >  to 10.255.225.5 via ge-0/0/1.1033
172.16.195.2/32     *[BGP/170] 00:14:35, localpref 100
                       AS path: 64911 I, validation-state: unverified
                    >  to 10.255.227.5 via ge-0/0/2.1033
192.168.230.0/24    *[Direct/0] 00:00:24
                    >  via ge-0/0/0.0
192.168.230.98/32   *[Local/0] 00:00:24
                       Local via ge-0/0/0.0
```

## Juniper SRX Series Firewall as WAN Router

The following example table and configurations show the differences between using an SRX Series Firewall in cluster mode and an MX router as the WAN router. On the fabric side, only the interface names of the SRX cluster change from the MX router configuration. Because the SRX Series Firewall runs in active/active cluster mode, there is only a single WAN router configuration and a single ASN to consider. That single configuration also includes cluster management and trust-zone management commands that are not present in a similar MX router-based configuration.

This SRX Series Firewall-based approach is less complicated than configuring redundant ethernet (reth) interfaces and link aggregation groups (LAG) on the MX router. In addition, there is need for additional CLI on the fabric side to insert virtual gateways, and so on.

### Preventing Potential Asymmetric Route Issues Using As-Path Prepending

When stateful firewalls are connected to a fabric, there's a potential risk of asymmetric routing. Consider the following scenario: a client outside the fabric initiates a TCP connection to a client inside the fabric via the WAN router. During the initial TCP three-way handshake, the SYN packet is handled by the first WAN router. However, the SYN-ACK response from the fabric client may be processed by the second WAN router, since both routers advertise the same default route into the fabric. The fabric, unaware of any preference, may choose the second WAN router due to ECMP load balancing or other internal mechanisms. If the stateful firewalls (acting as WAN routers) do not share session state, the second WAN router may drop the return packet, as it has no record of the session initiated through the first WAN router.

To prevent this issue, you can make the default route advertised by the second WAN router less preferred, ensuring that return traffic, under normal conditions, is directed to the first WAN router. In BGP, this is typically achieved through AS-Path prepending, which makes the route to the second WAN router less attractive. In the SRX cluster configuration shown below, we use two export filters toward

the fabric—one of which includes AS-Path prepending for the default route. These filters are then applied to the fabric neighbours based on which WAN router should be less preferred.

```
set policy-options policy-statement internet0 term 1 from protocol static
set policy-options policy-statement internet0 term 1 from route-filter 0.0.0.0/0 exact
set policy-options policy-statement internet0 term 1 then accept
set policy-options policy-statement internet0 term 2 then reject
set policy-options policy-statement internet1 term 1 from protocol static
set policy-options policy-statement internet1 term 1 from route-filter 0.0.0.0/0 exact
set policy-options policy-statement internet1 term 1 then as-path-prepend "64999 64999"
set policy-options policy-statement internet1 term 1 then accept
set policy-options policy-statement internet1 term 2 then reject
```

Once BGP peering with the fabric is established, you can verify the default routes received by the fabric. In the example below, core1-switch serves as the service block function.

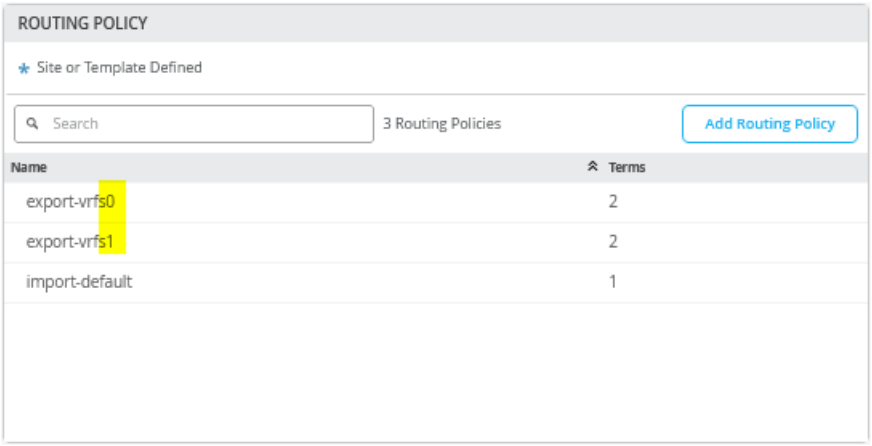
```
root@core1> show route aspath-regex ".* 64901 .*"
.
devices.inet.0: 13 destinations, 19 routes (13 active, 0 holddown, 0 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both
.
0.0.0.0/0          *[BGP/170] 00:00:24, localpref 100
                      AS path: 64901 I, validation-state: unverified
                    >  to 10.255.224.4 via ge-0/0/5.1031
                    [BGP/170] 00:00:24, localpref 100
                      AS path: 64999 64999 64901 I, validation-state: unverified
                    >  to 10.255.225.4 via ge-0/0/6.1033
.
customerb.inet.0: 14 destinations, 20 routes (14 active, 0 holddown, 0 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both
.
0.0.0.0/0          *[BGP/170] 00:00:24, localpref 100
                      AS path: 64901 I, validation-state: unverified
                    >  to 10.255.224.2 via ge-0/0/5.1081
                    [BGP/170] 00:00:24, localpref 100
                      AS path: 64999 64999 64901 I, validation-state: unverified
                    >  to 10.255.225.2 via ge-0/0/6.1088
.
customera.inet.0: 14 destinations, 20 routes (14 active, 0 holddown, 0 hidden)
```

```
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both

.
0.0.0.0/0          *[BGP/170] 00:00:24, localpref 100
                      AS path: 64901 I, validation-state: unverified
                    >  to 10.255.224.0 via ge-0/0/5.1091
                    [BGP/170] 00:00:24, localpref 100
                      AS path: 64999 64999 64901 I, validation-state: unverified
                    >  to 10.255.225.0 via ge-0/0/6.1099
.
```

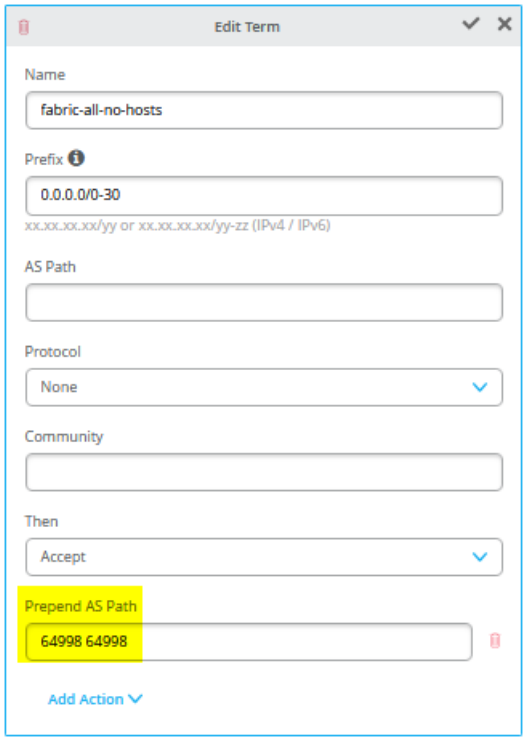
A similar approach is recommended for the fabric configuration to ensure uplink traffic consistently prefers one WAN router. This is achieved by making the routes advertised to the second WAN router less attractive using AS-Path prepending. Unlike the MX router example above, which uses a shared “export-vrfs” filter, we now use two separate filters—“export-vrfs0” and “export-vrfs1”—as shown in Figure 45 on page 85 .

**Figure 45: Two export filters instead of one**




They may look the same but all statements in “export-vrfs1” have AP-Path prepending configured as in Figure 46 on page 86.

**Figure 46: Second Filter Rules Always Contain AP-Path Prepending**




Once BGP peering with the SRX cluster is established, verify that the AS-Path prepends in the fabric are visible, as shown in the example below from the SRX cluster. Start by checking the configuration on the fabric's service block function—in this case, core1-switch.

```
root@core1> show configuration | display set | match export-vrf
set groups top policy-options policy-statement export-vrfs0 term 01_fabric-all-no-hosts from
route-filter-list 0-0-0-0_0-30
set groups top policy-options policy-statement export-vrfs0 term 01_fabric-all-no-hosts then
accept
set groups top policy-options policy-statement export-vrfs0 term 02_overlaylo0 from route-filter-
list 172-16-192-0_24-32
set groups top policy-options policy-statement export-vrfs0 term 02_overlaylo0 then accept
set groups top policy-options policy-statement export-vrfs1 term 01_fabric-all-no-hosts from
route-filter-list 0-0-0-0_0-30
set groups top policy-options policy-statement export-vrfs1 term 01_fabric-all-no-hosts then as-
path-prepend "64998 64998"
set groups top policy-options policy-statement export-vrfs1 term 01_fabric-all-no-hosts then
accept
set groups top policy-options policy-statement export-vrfs1 term 02_overlaylo0 from route-filter-
list 172-16-192-0_24-32
set groups top policy-options policy-statement export-vrfs1 term 02_overlaylo0 then as-path-
```

```
prepend "64998 64998"
set groups top policy-options policy-statement export-vrfs1 term 02_overlaylo0 then accept
set groups top routing-instances devices protocols bgp group devices0 export export-vrfs0
set groups top routing-instances devices protocols bgp group devices1 export export-vrfs1
set groups top routing-instances customerb protocols bgp group customerb0 export export-vrfs0
set groups top routing-instances customerb protocols bgp group customerb1 export export-vrfs1
set groups top routing-instances customera protocols bgp group customera0 export export-vrfs0
set groups top routing-instances customera protocols bgp group customera1 export export-vrfs1
```

Then, we also ensure that we receive those routes according to our configuration on the SRX cluster as in below example.

```
# neighbor-IP:10.255.224.1 which is assigned on srx-cluster to vrf:customera interface
ge-0/0/2.1091
root@srx1> show route receive-protocol bgp 10.255.224.1
.
public-int.inet.0: 46 destinations, 101 routes (46 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.99.91.0/24           10.255.224.1                            64911 65002 65004 65005 I
* 10.99.99.0/24           10.255.224.1                            64911 65002 65004 65005 I
* 172.16.192.1/32         10.255.224.1                            64911 65002 65004 65005 I
* 172.16.192.4/32         10.255.224.1                            64911 65002 65004 65006 I
* 172.16.192.7/32         10.255.224.1                            64911 I
  172.16.192.10/32        10.255.224.1                            64911 65002 65004 65001 I
.
# neighbor-IP:10.255.226.1 which is assigned on srx-cluster to vrf:customera interface
ge-0/0/3.1091
root@srx1> show route receive-protocol bgp 10.255.226.1
.
public-int.inet.0: 46 destinations, 101 routes (46 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
  10.99.91.0/24           10.255.226.1                            64911 65001 65004 65005 I
  10.99.99.0/24           10.255.226.1                            64911 65001 65004 65005 I
  172.16.192.1/32         10.255.226.1                            64911 65001 65004 65005 I
  172.16.192.4/32         10.255.226.1                            64911 65001 65004 65006 I
  172.16.192.7/32         10.255.226.1                            64911 65001 65004 65002 I
* 172.16.192.10/32        10.255.226.1                            64911 I
.
# neighbor-IP:10.255.225.1 which is assigned on srx-cluster to vrf:customera interface
ge-7/0/2.1099
root@srx1> show route receive-protocol bgp 10.255.225.1
```

```
.
public-int.inet.0: 46 destinations, 101 routes (46 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
  10.99.91.0/24           10.255.225.1                            64998 64998 64911 65002 65004
65005 I
  10.99.99.0/24           10.255.225.1                            64998 64998 64911 65002 65004
65005 I
  172.16.192.1/32         10.255.225.1                            64998 64998 64911 65002 65004
65005 I
  172.16.192.4/32         10.255.225.1                            64998 64998 64911 65002 65004
65006 I
  172.16.192.7/32         10.255.225.1                            64998 64998 64911 I
  172.16.192.10/32        10.255.225.1                            64998 64998 64911 65002 65004
65001 I
.
# neighbor-IP:10.255.227.1 which is assigned on srx-cluster to vrf:customera interface
ge-7/0/3.1099
root@srx1> show route receive-protocol bgp 10.255.227.1
.
public-int.inet.0: 46 destinations, 101 routes (46 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
  10.99.91.0/24           10.255.227.1                            64998 64998 64911 65001 65004
65005 I
  10.99.99.0/24           10.255.227.1                            64998 64998 64911 65001 65004
65005 I
  172.16.192.1/32         10.255.227.1                            64998 64998 64911 65001 65004
65005 I
  172.16.192.4/32         10.255.227.1                            64998 64998 64911 65001 65004
65006 I
  172.16.192.7/32         10.255.227.1                            64998 64998 64911 65001 65004
65002 I
  172.16.192.10/32        10.255.227.1                            64998 64998 64911 I
```

Table 2 on page 89 shows the configuration information for the core1 and core2 switches as service block function along with the WAN router configuration for the SRX cluster. We've marked the changes with respect to Table 1 on page 52 (for MX WAN routers in bold).

**Table 2: SRX Series Firewall WAN Router and Core Switch Configuration Summary**

| Switch | Switch AS | VRF | Core P2P IP | Core IF | WAN Router | WAN Router P2P IP | WAN Router AS | WAN Router IF | VLAN-ID |
|---|---|---|---|---|---|---|---|---|---|
| core1 | 64911 | customera | 10.255.224.1/31 | ge-0/0/**5**.1091 | **node0** | 10.255.224.0/31 | 64901 | ge-0/0/**2**.1091 | 1091 |
| core1 | 64911 | customerb | 10.255.224.3/31 | ge-0/0/**5**.1081 | **node0** | 10.255.224.2/31 | 64901 | ge-0/0/**2**.1081 | 1081 |
| core1 | 64911 | devices | 10.255.224.5/31 | ge-0/0/**5**.1031 | **node0** | 10.255.224.4/31 | 64901 | ge-0/0/**2**.1031 | 1031 |
| core1 | 64911 | customera | 10.255.225.1/31 | ge-0/0/**6**.1099 | **node1** | 10.255.225.0/31 | 64901 | ge-**7**/0/**2**.1099 | 1099 |
| core1 | 64911 | customerb | 10.255.225.3/31 | ge-0/0/**6**.1088 | **node1** | 10.255.225.2/31 | 64901 | ge-**7**/0/**2**.1088 | 1088 |
| core1 | 64911 | devices | 10.255.225.5/31 | ge-0/0/**6**.1033 | **node1** | 10.255.225.4/31 | 64901 | ge-**7**/0/**2**.1033 | 1033 |
| core2 | 64911 | customera | 10.255.226.1/31 | ge-0/0/**5**.1091 | **node0** | 10.255.226.0/31 | 64901 | ge-0/0/**3**.1091 | 1091 |
| core2 | 64911 | customerb | 10.255.226.3/31 | ge-0/0/**5**.1081 | **node0** | 10.255.226.2/31 | 64901 | ge-0/0/**3**.1081 | 1081 |
| core2 | 64911 | devices | 10.255.226.5/31 | ge-0/0/**5**.1031 | **node0** | 10.255.226.4/31 | 64901 | ge-0/0/**3**.1031 | 1031 |
| core2 | 64911 | customera | 10.255.227.1/31 | ge-0/0/**6**.1099 | **node1** | 10.255.227.0/31 | 64901 | ge-**7**/0/**3**.1099 | 1099 |
| core2 | 64911 | customerb | 10.255.227.3/31 | ge-0/0/**6**.1088 | **node1** | 10.255.227.2/31 | 64901 | ge-**7**/0/**3**.1088 | 1088 |
| core2 | 64911 | devices | 10.255.227.5/31 | ge-0/0/**6**.1033 | **node1** | 10.255.227.4/31 | 64901 | ge-**7**/0/**3**.1033 | 1033 |

The following pseudocode describes what you need to configure on the **core1** switch for this example:

```
# configure the Additional IP-Subnet 10.255.224.1 255.255.255.254 to Network/VLAN:vlan1091
# configure the Additional IP-Subnet 10.255.224.3 255.255.255.254 to Network/VLAN:vlan1081
# configure the Additional IP-Subnet 10.255.224.5 255.255.255.254 to Network/VLAN:vlan1031
# Then bind these 3 Network/VLANs to Port Interface ge-0/0/5 as L3-Sub-Interfaces with MTU=9018
#
# configure the Additional IP-Subnet 10.255.225.1 255.255.255.254 to Network/VLAN:vlan1099
# configure the Additional IP-Subnet 10.255.225.3 255.255.255.254 to Network/VLAN:vlan1088
# configure the Additional IP-Subnet 10.255.225.5 255.255.255.254 to Network/VLAN:vlan1033
# Then bind these 3 Network/VLANs to Port Interface ge-0/0/6 as L3-Sub-Interfaces with MTU=9018
#
# Enable BGP
# Create an Export policy called 'export-vrfs0'
# Add to this export Policy the following Networks as:
# - Add Term w. Name=fabric-all-no-hosts  Prefix=0.0.0.0/0-30  Protocol=None  Then=Accept
# - Add Term w. Name=overlaylo0 Prefix=172.16.192.0/24-32  Protocol=None  Then=Accept
#
# Create an Export policy called 'export-vrfs1'
# Add to this export Policy the following Networks as:
# - Add Term w. Name=fabric-all-no-hosts  Prefix=0.0.0.0/0-30  Protocol=None  Then=Accept  Add
Action->Prepend AS Path='64998 64998'
# - Add Term w. Name=overlaylo0 Prefix=172.16.192.0/24-32  Protocol=None  Then=Accept  Add
Action->Prepend AS Path='64998 64998'
#
# Create an Export policy called 'import-default'
# - Name=default  Prefix=0.0.0.0/0  Protocol=BGP  Action=Accept
#
# Create a BGP Group with:
# - Name=customera0
# - Type=External
# - Network (VLAN)=vlan1091
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs0 and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.224.0  Neighbor_AS=64901  Hold-Time=90
#
# Create a BGP Group with:
# - Name=customerb0
# - Type=External
```

```
# - Network (VLAN)=vlan1081
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs0 and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.224.2  Neighbor_AS=64901  Hold-Time=90
#
# Create a BGP Group with:
# - Name=devices0
# - Type=External
# - Network (VLAN)=vlan1031
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs0 and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.224.4  Neighbor_AS=64901  Hold-Time=90
#
# Create a BGP Group with:
# - Name=customera1
# - Type=External
# - Network (VLAN)=vlan1099
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs1 and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.225.0  Neighbor_AS=64901  Hold-Time=90
#
# Create a BGP Group with:
# - Name=customerb1
# - Type=External
# - Network (VLAN)=vlan1088
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs1 and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.225.2  Neighbor_AS=64901  Hold-Time=90
#
# Create a BGP Group with:
# - Name=devices1
```

```
# - Type=External
# - Network (VLAN)=vlan1033
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs1 and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.225.4  Neighbor_AS=64901  Hold-Time=90
```

The following pseudocode describes what you need to configure on the **core2** switch for this example:

```
# configure the Additional IP-Subnet 10.255.226.1 255.255.255.254 to Network/VLAN:vlan1091
# configure the Additional IP-Subnet 10.255.226.3 255.255.255.254 to Network/VLAN:vlan1081
# configure the Additional IP-Subnet 10.255.226.5 255.255.255.254 to Network/VLAN:vlan1031
# Then bind these 3 Network/VLANs to Port Interface ge-0/0/5 as L3-Sub-Interfaces with MTU=9018
#
# configure the Additional IP-Subnet 10.255.227.1 255.255.255.254 to Network/VLAN:vlan1099
# configure the Additional IP-Subnet 10.255.227.3 255.255.255.254 to Network/VLAN:vlan1088
# configure the Additional IP-Subnet 10.255.227.5 255.255.255.254 to Network/VLAN:vlan1033
# Then bind these 3 Network/VLANs to Port Interface ge-0/0/6 as L3-Sub-Interfaces with MTU=9018
#
# Enable BGP
# Create an Export policy called 'export-vrfs0'
# Add to this export Policy the following Networks as:
# - Add Term w. Name=fabric-all-no-hosts  Prefix=0.0.0.0/0-30  Protocol=None  Then=Accept
# - Add Term w. Name=overlaylo0 Prefix=172.16.192.0/24-32  Protocol=None  Then=Accept
#
# Create an Export policy called 'export-vrfs1'
# Add to this export Policy the following Networks as:
# - Add Term w. Name=fabric-all-no-hosts  Prefix=0.0.0.0/0-30  Protocol=None  Then=Accept  Add
Action->Prepend AS Path='64998 64998'
# - Add Term w. Name=overlaylo0 Prefix=172.16.192.0/24-32  Protocol=None  Then=Accept  Add
Action->Prepend AS Path='64998 64998'
#
# Create an Export policy called 'import-default'
# - Name=default  Prefix=0.0.0.0/0  Protocol=BGP  Action=Accept
#
# Create a BGP Group with:
# - Name=customera0
# - Type=External
# - Network (VLAN)=vlan1091
# - BFD interval=1000
```

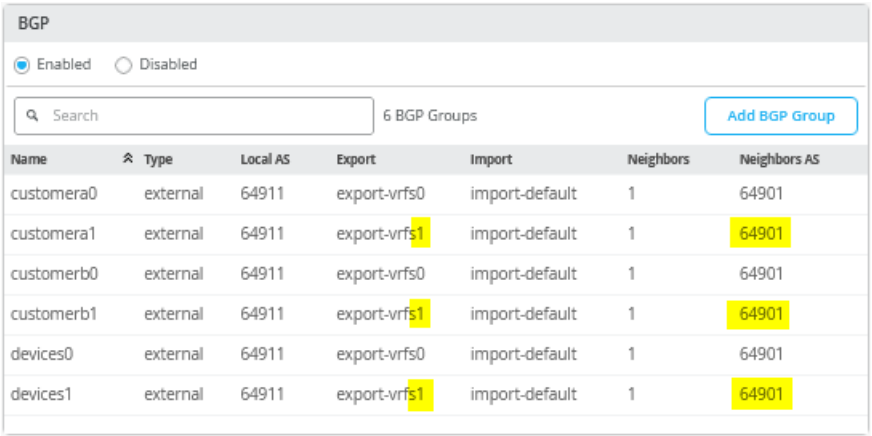
```
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs0 and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.226.0  Neighbor_AS=64901  Hold-Time=90
#
# Create a BGP Group with:
# - Name=customerb0
# - Type=External
# - Network (VLAN)=vlan1081
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs0 and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.226.2  Neighbor_AS=64901  Hold-Time=90
#
# Create a BGP Group with:
# - Name=devices0
# - Type=External
# - Network (VLAN)=vlan1031
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs0 and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.226.4  Neighbor_AS=64901  Hold-Time=90
#
# Create a BGP Group with:
# - Name=customera1
# - Type=External
# - Network (VLAN)=vlan1099
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs1 and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.227.0  Neighbor_AS=64901  Hold-Time=90
#
# Create a BGP Group with:
# - Name=customerb1
# - Type=External
# - Network (VLAN)=vlan1088
```

```
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs1 and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.227.2  Neighbor_AS=64901  Hold-Time=90
#
# Create a BGP Group with:
# - Name=devices1
# - Type=External
# - Network (VLAN)=vlan1033
# - BFD interval=1000
# - Local AS=64911
# - Hold Time=90
# - Set Export=export-vrfs1 and Import=import-default
# Add also the following Neighbor
# - IP_Address=10.255.227.4  Neighbor_AS=64901  Hold-Time=90
```

When finished with configuring the individual service block functions (here, the core1 and core2 switches) your overview table should be as shown in Figure 47 on page 94 .

Figure 47 on page 94 shows an overview of how the BGP looks after you have configured the individual service block functions for the core1 and core2 switches.

**Figure 47: BGP Configuration Summary with SRX Series Firewall as WAN Router**



The following Junos OS CLI represents the entire configuration needed on the Series Firewall cluster for this example.

```
set groups node0 system host-name srx1_node0
set groups node1 system host-name srx2_node1
set apply-groups ""
set chassis aggregated-devices ethernet device-count 10
set chassis cluster control-link-recovery
set chassis cluster reth-count 5
set chassis cluster initial-hold 60
set chassis cluster redundancy-group 1 node 0 priority 200
set chassis cluster redundancy-group 1 node 1 priority 100
set chassis cluster redundancy-group 1 gratuitous-arp-count 4
# Interface monitoring turned OFF for vSRX
# set chassis cluster redundancy-group 1 interface-monitor ge-0/0/2 weight 255
# set chassis cluster redundancy-group 1 interface-monitor ge-0/0/3 weight 255
# set chassis cluster redundancy-group 1 interface-monitor ge-7/0/2 weight 255
# set chassis cluster redundancy-group 1 interface-monitor ge-7/0/3 weight 255
set interfaces fab0 fabric-options member-interfaces ge-0/0/0
set interfaces fab1 fabric-options member-interfaces ge-7/0/0
#
# rebuild trust zone to clear old interfaces bound
delete security zones security-zone trust
set security zones security-zone trust tcp-rst
set security zones security-zone trust host-inbound-traffic system-services all
set security zones security-zone trust host-inbound-traffic protocols all
#
delete interfaces ge-0/0/2
set interfaces ge-0/0/2 flexible-vlan-tagging
set interfaces ge-0/0/2 mtu 9018
set interfaces ge-0/0/2 unit 1091 description vlan1091
set interfaces ge-0/0/2 unit 1091 vlan-id 1091
set interfaces ge-0/0/2 unit 1091 family inet address 10.255.224.0/31
set security zones security-zone trust interfaces ge-0/0/2.1091 host-inbound-traffic protocols
bgp
set security zones security-zone trust interfaces ge-0/0/2.1091 host-inbound-traffic system-
services ping
set interfaces ge-0/0/2 unit 1081 description vlan1081
set interfaces ge-0/0/2 unit 1081 vlan-id 1081
set interfaces ge-0/0/2 unit 1081 family inet address 10.255.224.2/31
set security zones security-zone trust interfaces ge-0/0/2.1081 host-inbound-traffic protocols
bgp
```

```
set security zones security-zone trust interfaces ge-0/0/2.1081 host-inbound-traffic system-
services ping
set interfaces ge-0/0/2 unit 1031 description vlan1031
set interfaces ge-0/0/2 unit 1031 vlan-id 1031
set interfaces ge-0/0/2 unit 1031 family inet address 10.255.224.4/31
set security zones security-zone trust interfaces ge-0/0/2.1031 host-inbound-traffic protocols
bgp
set security zones security-zone trust interfaces ge-0/0/2.1031 host-inbound-traffic system-
services ping
#
delete interfaces ge-0/0/3
set interfaces ge-0/0/3 flexible-vlan-tagging
set interfaces ge-0/0/3 mtu 9018
set interfaces ge-0/0/3 unit 1091 description vlan1091
set interfaces ge-0/0/3 unit 1091 vlan-id 1091
set interfaces ge-0/0/3 unit 1091 family inet address 10.255.226.0/31
set security zones security-zone trust interfaces ge-0/0/3.1091 host-inbound-traffic protocols
bgp
set security zones security-zone trust interfaces ge-0/0/3.1091 host-inbound-traffic system-
services ping
set interfaces ge-0/0/3 unit 1081 description vlan1081
set interfaces ge-0/0/3 unit 1081 vlan-id 1081
set interfaces ge-0/0/3 unit 1081 family inet address 10.255.226.2/31
set security zones security-zone trust interfaces ge-0/0/3.1081 host-inbound-traffic protocols
bgp
set security zones security-zone trust interfaces ge-0/0/3.1081 host-inbound-traffic system-
services ping
set interfaces ge-0/0/3 unit 1031 description vlan1031
set interfaces ge-0/0/3 unit 1031 vlan-id 1031
set interfaces ge-0/0/3 unit 1031 family inet address 10.255.226.4/31
set security zones security-zone trust interfaces ge-0/0/3.1031 host-inbound-traffic protocols
bgp
set security zones security-zone trust interfaces ge-0/0/3.1031 host-inbound-traffic system-
services ping
#
delete interfaces ge-7/0/2
set interfaces ge-7/0/2 flexible-vlan-tagging
set interfaces ge-7/0/2 mtu 9018
set interfaces ge-7/0/2 unit 1099 description vlan1099
set interfaces ge-7/0/2 unit 1099 vlan-id 1099
set interfaces ge-7/0/2 unit 1099 family inet address 10.255.225.0/31
set security zones security-zone trust interfaces ge-7/0/2.1099 host-inbound-traffic protocols
bgp
```

```
set security zones security-zone trust interfaces ge-7/0/2.1099 host-inbound-traffic system-
services ping
set interfaces ge-7/0/2 unit 1088 description vlan1088
set interfaces ge-7/0/2 unit 1088 vlan-id 1088
set interfaces ge-7/0/2 unit 1088 family inet address 10.255.225.2/31
set security zones security-zone trust interfaces ge-7/0/2.1088 host-inbound-traffic protocols
bgp
set security zones security-zone trust interfaces ge-7/0/2.1088 host-inbound-traffic system-
services ping
set interfaces ge-7/0/2 unit 1033 description vlan1033
set interfaces ge-7/0/2 unit 1033 vlan-id 1033
set interfaces ge-7/0/2 unit 1033 family inet address 10.255.225.4/31
set security zones security-zone trust interfaces ge-7/0/2.1033 host-inbound-traffic protocols
bgp
set security zones security-zone trust interfaces ge-7/0/2.1033 host-inbound-traffic system-
services ping
#
delete interfaces ge-7/0/3
set interfaces ge-7/0/3 flexible-vlan-tagging
set interfaces ge-7/0/3 mtu 9018
set interfaces ge-7/0/3 unit 1099 description vlan1099
set interfaces ge-7/0/3 unit 1099 vlan-id 1099
set interfaces ge-7/0/3 unit 1099 family inet address 10.255.227.0/31
set security zones security-zone trust interfaces ge-7/0/3.1099 host-inbound-traffic protocols
bgp
set security zones security-zone trust interfaces ge-7/0/3.1099 host-inbound-traffic system-
services ping
set interfaces ge-7/0/3 unit 1088 description vlan1088
set interfaces ge-7/0/3 unit 1088 vlan-id 1088
set interfaces ge-7/0/3 unit 1088 family inet address 10.255.227.2/31
set security zones security-zone trust interfaces ge-7/0/3.1088 host-inbound-traffic protocols
bgp
set security zones security-zone trust interfaces ge-7/0/3.1088 host-inbound-traffic system-
services ping
set interfaces ge-7/0/3 unit 1033 description vlan1033
set interfaces ge-7/0/3 unit 1033 vlan-id 1033
set interfaces ge-7/0/3 unit 1033 family inet address 10.255.227.4/31
set security zones security-zone trust interfaces ge-7/0/3.1033 host-inbound-traffic protocols
bgp
set security zones security-zone trust interfaces ge-7/0/3.1033 host-inbound-traffic system-
services ping
#
# needed in and export policy
```

```
delete policy-options
set policy-options policy-statement fabric term 1 from protocol bgp
set policy-options policy-statement fabric term 1 from route-filter 0.0.0.0/0 orlonger
set policy-options policy-statement fabric term 1 then accept
set policy-options policy-statement fabric term 2 then reject
set policy-options policy-statement internet0 term 1 from protocol static
set policy-options policy-statement internet0 term 1 from route-filter 0.0.0.0/0 exact
set policy-options policy-statement internet0 term 1 then accept
set policy-options policy-statement internet0 term 2 then reject
set policy-options policy-statement internet1 term 1 from protocol static
set policy-options policy-statement internet1 term 1 from route-filter 0.0.0.0/0 exact
set policy-options policy-statement internet1 term 1 then as-path-prepend "64999 64999"
set policy-options policy-statement internet1 term 1 then accept
set policy-options policy-statement internet1 term 2 then reject
#
delete routing-instances public-int
set routing-instances public-int instance-type virtual-router
set routing-instances public-int interface ge-0/0/2.1091
set routing-instances public-int interface ge-0/0/2.1081
set routing-instances public-int interface ge-0/0/2.1031
set routing-instances public-int interface ge-0/0/3.1091
set routing-instances public-int interface ge-0/0/3.1081
set routing-instances public-int interface ge-0/0/3.1031
set routing-instances public-int interface ge-7/0/2.1099
set routing-instances public-int interface ge-7/0/2.1088
set routing-instances public-int interface ge-7/0/2.1033
set routing-instances public-int interface ge-7/0/3.1099
set routing-instances public-int interface ge-7/0/3.1088
set routing-instances public-int interface ge-7/0/3.1033
#
delete routing-instances public-int protocols bgp group customera
delete routing-instances public-int protocols bgp group customera0
set routing-instances public-int protocols bgp group customera0 type external
set routing-instances public-int protocols bgp group customera0 family inet unicast
set routing-instances public-int protocols bgp group customera0 multipath multiple-as
set routing-instances public-int protocols bgp group customera0 local-as 64901
set routing-instances public-int protocols bgp group customera0 hold-time 90
set routing-instances public-int protocols bgp group customera0 import fabric
set routing-instances public-int protocols bgp group customera0 export internet0
set routing-instances public-int protocols bgp group customera0 bfd-liveness-detection minimum-
interval 1000
set routing-instances public-int protocols bgp group customera0 bfd-liveness-detection
multiplier 3
```

```
set routing-instances public-int protocols bgp group customera0 bfd-liveness-detection session-
mode automatic
set routing-instances public-int protocols bgp group customera0 neighbor 10.255.224.1 peer-as
64911
set routing-instances public-int protocols bgp group customera0 neighbor 10.255.226.1 peer-as
64911
#
delete routing-instances public-int protocols bgp group customera1
set routing-instances public-int protocols bgp group customera1 type external
set routing-instances public-int protocols bgp group customera1 family inet unicast
set routing-instances public-int protocols bgp group customera1 multipath multiple-as
set routing-instances public-int protocols bgp group customera1 local-as 64901
set routing-instances public-int protocols bgp group customera1 hold-time 90
set routing-instances public-int protocols bgp group customera1 import fabric
set routing-instances public-int protocols bgp group customera1 export internet1
set routing-instances public-int protocols bgp group customera1 bfd-liveness-detection minimum-
interval 1000
set routing-instances public-int protocols bgp group customera1 bfd-liveness-detection
multiplier 3
set routing-instances public-int protocols bgp group customera1 bfd-liveness-detection session-
mode automatic
set routing-instances public-int protocols bgp group customera1 neighbor 10.255.225.1 peer-as
64911
set routing-instances public-int protocols bgp group customera1 neighbor 10.255.227.1 peer-as
64911
#
delete routing-instances public-int protocols bgp group customerb
delete routing-instances public-int protocols bgp group customerb0
set routing-instances public-int protocols bgp group customerb0 type external
set routing-instances public-int protocols bgp group customerb0 family inet unicast
set routing-instances public-int protocols bgp group customerb0 multipath multiple-as
set routing-instances public-int protocols bgp group customerb0 local-as 64901
set routing-instances public-int protocols bgp group customerb0 hold-time 90
set routing-instances public-int protocols bgp group customerb0 import fabric
set routing-instances public-int protocols bgp group customerb0 export internet0
set routing-instances public-int protocols bgp group customerb0 bfd-liveness-detection minimum-
interval 1000
set routing-instances public-int protocols bgp group customerb0 bfd-liveness-detection
multiplier 3
set routing-instances public-int protocols bgp group customerb0 bfd-liveness-detection session-
mode automatic
set routing-instances public-int protocols bgp group customerb0 neighbor 10.255.224.3 peer-as
64911
```

```
set routing-instances public-int protocols bgp group customerb0 neighbor 10.255.226.3 peer-as
64911
#
delete routing-instances public-int protocols bgp group customerb1
set routing-instances public-int protocols bgp group customerb1 type external
set routing-instances public-int protocols bgp group customerb1 family inet unicast
set routing-instances public-int protocols bgp group customerb1 multipath multiple-as
set routing-instances public-int protocols bgp group customerb1 local-as 64901
set routing-instances public-int protocols bgp group customerb1 hold-time 90
set routing-instances public-int protocols bgp group customerb1 import fabric
set routing-instances public-int protocols bgp group customerb1 export internet1
set routing-instances public-int protocols bgp group customerb1 bfd-liveness-detection minimum-
interval 1000
set routing-instances public-int protocols bgp group customerb1 bfd-liveness-detection
multiplier 3
set routing-instances public-int protocols bgp group customerb1 bfd-liveness-detection session-
mode automatic
set routing-instances public-int protocols bgp group customerb1 neighbor 10.255.225.3 peer-as
64911
set routing-instances public-int protocols bgp group customerb1 neighbor 10.255.227.3 peer-as
64911
#
delete routing-instances public-int protocols bgp group devices
delete routing-instances public-int protocols bgp group devices0
set routing-instances public-int protocols bgp group devices0 type external
set routing-instances public-int protocols bgp group devices0 family inet unicast
set routing-instances public-int protocols bgp group devices0 multipath multiple-as
set routing-instances public-int protocols bgp group devices0 local-as 64901
set routing-instances public-int protocols bgp group devices0 hold-time 90
set routing-instances public-int protocols bgp group devices0 import fabric
set routing-instances public-int protocols bgp group devices0 export internet0
set routing-instances public-int protocols bgp group devices0 bfd-liveness-detection minimum-
interval 1000
set routing-instances public-int protocols bgp group devices0 bfd-liveness-detection multiplier 3
set routing-instances public-int protocols bgp group devices0 bfd-liveness-detection session-
mode automatic
set routing-instances public-int protocols bgp group devices0 neighbor 10.255.224.5 peer-as 64911
set routing-instances public-int protocols bgp group devices0 neighbor 10.255.226.5 peer-as 64911
#
delete routing-instances public-int protocols bgp group devices1
set routing-instances public-int protocols bgp group devices1 type external
set routing-instances public-int protocols bgp group devices1 family inet unicast
set routing-instances public-int protocols bgp group devices1 multipath multiple-as
```

```
set routing-instances public-int protocols bgp group devices1 local-as 64901
set routing-instances public-int protocols bgp group devices1 hold-time 90
set routing-instances public-int protocols bgp group devices1 import fabric
set routing-instances public-int protocols bgp group devices1 export internet1
set routing-instances public-int protocols bgp group devices1 bfd-liveness-detection minimum-
interval 1000
set routing-instances public-int protocols bgp group devices1 bfd-liveness-detection multiplier 3
set routing-instances public-int protocols bgp group devices1 bfd-liveness-detection session-
mode automatic
set routing-instances public-int protocols bgp group devices1 neighbor 10.255.225.5 peer-as 64911
set routing-instances public-int protocols bgp group devices1 neighbor 10.255.227.5 peer-as 64911
```