JUNIPER NETWORKS | Engineering Simplicity

Junos® OS

Layer 2 Bridging, Address Learning, and Forwarding User Guide

Published
2024-12-19

JUNOS

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Junos® OS Layer 2 Bridging, Address Learning, and Forwarding User Guide

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at https://support.juniper.net/support/eula/. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

About This Guide

Use this guide to configure, monitor, and troubleshoot Layer 2 bridging, address learning, and forwarding features on your Juniper Network devices.

1
CHAPTER

Layer 2 Bridging

Layer 2 Bridge Domains Overview

SUMMARY

IN THIS SECTION

Understanding Layer 2 Bridge Domains on MX Series

You can configure one or more bridge domains on MX Series routers to perform Layer 2 bridging. The Layer 2 bridging functions of the MX Series routers include integrated routing and bridging (IRB) for support for Layer 2 bridging and Layer 3 IP routing on the same interface, and virtual switches that isolate a LAN segment with its spanning-tree protocol instance and separate its VLAN ID space.

A bridge domain is a set of logical ports that share the same flooding or broadcast characteristics. Like a virtual LAN (VLAN), a bridge domain spans one or more ports of multiple devices.

On Juniper Networks MX Series 5G Universal Routing Platforms only, you can configure one or more bridge domains to perform Layer 2 bridging. Thus, MX Series routers can function as Layer 2 switches, each with multiple bridging, or broadcast, domains that participate in the same Layer 2 network. You can also configure Layer 3 routing support for a bridge domain. Integrated routing and bridging (IRB) provides support for Layer 2 bridging and Layer 3 IP routing on the same interface. IRB enables you to route packets to another routed interface or to another bridge domain that has a Layer 3 protocol configured.

You can also group one or more bridge domains within a single instance, or virtual switch. The MX Series routers also support multiple virtual switches, each of which operates independently of other virtual switches on the router. Virtual switches isolate a LAN segment with its spanning-tree protocol instance. . Thus, each virtual switch can participate in a different Layer 2 network.

In Junos OS Release 9.2 and later, bridge domains provide support for a Layer 2 trunk port. A Layer 2 trunk interface enables you to configure a single *logical interface* to represent multiple VLANs on a physical interface. You can configure a set of bridge domains and VLAN identifiers that are automatically associated with one or more Layer 2 trunk interfaces. Packets received on a trunk interface are forwarded within a bridge domain that has the same VLAN identifier. A Layer 2 trunk interface also

supports IRB within a bridge domain. In addition, you can configure Layer 2 learning and forwarding properties that apply to the entire set of bridge domains.

In Junos OS Release 9.3 and later, you can configure VPLS ports in a virtual switch instead of a dedicated routing instance of type **vpls** so that the logical interfaces of the Layer 2 bridge domains in the virtual switch can handle VPLS routing instance traffic. Packets received on a Layer 2 trunk interface are forwarded within a bridge domain that has the same VLAN identifier.

Understanding Layer 2 Bridge Domains on ACX Series

A bridge domain is a set of logical interfaces that share the same flooding or broadcast characteristics. Layer 2 logical interfaces are created by defining one or more logical units on a physical interface with encapsulation as `ethernet-bridge` or `vlan-bridge`. All the member ports of the bridge domain participate in Layer 2 learning and forwarding. You can configure one or more bridge domains on ACX Series routers to perform Layer 2 bridging. The Layer 2 bridging functions of ACX Series routers include integrated routing and bridging (IRB) support for Layer 2 bridging and Layer 3 IP routing on the same interface. IRB enables you to route packets to another routed interface or to another bridge domain that has a Layer 3 protocol configured

NOTE: ACX Series routers do not support the creation of bridge domains by using access and trunk ports.

You can configure E-LAN and E-LINE services by using bridge domains.

On ACX Series routers, you can configure bridge domains by using the following methods:

- Bridge domain without a `vlan-id` *`number`* statement
- Bridge domain with the `vlan-id` value set to *none*
- Bridge domain with a single `vlan-id`
- Bridge domain with a `vlan-id-list`

NOTE: The Layer 2 CLI configurations and show commands for ACX5048 and ACX5096 routers differ compared to other ACX Series routers. For more information, see Layer 2 Next Generation Mode for ACX Series.

When you configure E-LAN and E-LINE services using a bridge domain without a `vlan-id` *`number`* statement, the bridge domain should explicitly be normalized to a service VLAN ID and TPID by configuring an input VLAN map under a logical interface. Explicit normalization is required when a

logical interface's outer VLAN ID and TPID is not the same as the service VLAN ID and TPID of the service being configured using a bridge domain.

The following input VLAN map functions are supported in ACX Series routers:

- `push`—Add a new VLAN tag to the top of the VLAN stack.
- `swap`—Replace the outer VLAN tag of the VLAN stack in a frame.
- `pop`—Remove a VLAN tag from the top of the VLAN tag stack.
- `swap-swap`—Replace both the outer and inner VLAN tags of the frame.
- `push-push`—Push two VLAN tags on top of the VLAN stack.

> **NOTE**: `push-push` does not work on ACX Series routers if the incoming packet already has a VLAN tag.

The following VLAN map functions are not supported in ACX Series routers:

- `swap-push`—Replace the outer VLAN tag of the frame and add a new VLAN tag to the top of the VLAN stack.
- `pop-swap`—Remove the outer VLAN tag of the frame and replace the inner VLAN tag of the frame.
- `pop-pop`—Remove both the outer and inner VLAN tags of the frame.

> **NOTE**: You can configure Q-in-Q tunneling by explicitly configuring an input VLAN map with the `push` function on the ingress logical interface.

A bridge domain can also be created by using aggregated Ethernet interfaces. Aggregated Ethernet interfaces are considered as logical interfaces in a bridge domain.

The following steps outline the process for bridging a packet received over a Layer 2 logical interface:

1. When a packet is received on a physical port, it is accepted only if the VLAN identifier of the packet matches the VLAN identifier of one of the logical interfaces configured on that port.
2. If the bridge domain is configured without a `vlan-id` *`number`* statement, then the VLAN tags are rewritten based on the input VLAN map configured on the logical interface and normalized to a service VLAN ID.
3. If the bridge domain is configured with a normalizing VLAN identifier by using the **vlan-id** ***number*** statement, the VLAN tags of the received packet are compared with the normalizing VLAN identifier.

If the VLAN tags of the packet are different from the normalizing VLAN identifier, the VLAN tags are rewritten as described in Table 1 on page 5.

4. If the source MAC address of the received packet is not present in the source MAC table, it is learned based on the normalizing VLAN identifier.

5. The packet is then forwarded toward one or more outbound Layer 2 logical interfaces based on the destination MAC address. A packet with a known unicast destination MAC address is forwarded only to one outbound logical interface.

6. If the bridge domain is configured without a `vlan-id` *number* statement, then for each outbound Layer 2 logical interface, the VLAN tags are rewritten based on the output VLAN map configured on that logical interface.

7. If the bridge domain is configured with a normalizing VLAN identifier by using the **vlan-id** ***number*** statement, for each outbound Layer 2 logical interface, the normalizing VLAN identifier configured for the bridge domain is compared with the VLAN tags configured on that logical interface. If the VLAN tags associated with an outbound logical interface do not match the normalizing VLAN identifier configured for the bridge domain, the VLAN tags are rewritten as described in Table 2 on page 6.

Table 1 on page 5 shows specific examples of how the VLAN tags of packets sent to the bridge domain are processed and translated, depending on your configuration. "–" means that the statement is not supported for the specified logical interface VLAN identifier. "No operation" means that the VLAN tags of the received packet are not translated for the specified input logical interface.

Table 1: Statement Usage and Input Rewrite Operations for VLAN Identifiers for a Bridge Domain

VLAN Identifier of Logical Interface	VLAN Configurations for Bridge Domain	
	vlan-id none	vlan-id 200
none	No operation	push 200
200	pop 200	No operation
1000	pop 1000	swap 1000 to 200
vlan-tags outer 2000 inner 300	pop 2000, pop 300	pop 2000, swap 300 to 200

Table 1: Statement Usage and Input Rewrite Operations for VLAN Identifiers for a Bridge Domain *(Continued)*

VLAN Identifier of Logical Interface	VLAN Configurations for Bridge Domain	
	vlan-id none	vlan-id 200
vlan-tags outer 100 inner 400	pop 100, pop 400	pop 100, swap 400 to 200
vlan-id-range 10-100	–	–

Table 2 on page 6 shows specific examples of how the VLAN tags for packets sent from the bridge domain are processed and translated, depending on your configuration. "–" means that the statement is not supported for the specified logical interface VLAN identifier. "No operation" means that the VLAN tags of the outbound packet are not translated for the specified output logical interface.

Table 2: Statement Usage and Output Rewrite Operations for VLAN Identifiers for a Bridge Domain

VLAN Identifier of Logical Interface	VLAN Configurations for Bridge Domain	
	vlan-id none	vlan-id 200
none	no operation	pop 200
200	push 200	No operation
1000	push 1000	swap 200 to 1000
vlan-tags outer 2000 inner 300	push 2000, push 300	swap 200 to 300, push 2000
vlan-tags outer 100 inner 400	push 100, push 400	swap 200 to 400, push 100
vlan-id-range 10-100	–	–

Limitations on Layer 2 bridging—The following Layer 2 bridging limitations apply for ACX Series Universal Metro Routers:

- A bridge domain cannot have two or more logical interfaces that belong to the same physical interface.
- A bridge domain with dual VLAN ID tag is not supported.
- The maximum number of supported input VLAN maps with TPID `swap` is 64.
- MAC learning cannot be disabled at a logical interface level.
- MAC limit per logical interface cannot be configured.

Configure Layer 2 Bridging

SUMMARY

IN THIS SECTION

Configuring a Bridge Domain

A bridge domain must include a set of logical interfaces that participate in Layer 2 learning and forwarding. You can optionally configure a VLAN identifier and a routing interface for the bridge domain to also support Layer 3 IP routing.

To enable a bridge domain, include the following statements:

```
[edit]
bridge-domains {
    bridge-domain-name {
        domain-type bridge:
        interface interface-name;
        routing-interface routing-interface-name;
        vlan-id (none | all | number);
        vlan-id-list [ vlan-id-numbers ];
        vlan-tags outer number inner number);
    }
}
```

NOTE: The Layer 2 CLI configurations and show commands for ACX5048 and ACX5096 routers differ compared to other ACX Series routers. For more information, see *Layer 2 Next Generation Mode for ACX Series*.

You cannot use the slash (/) character in bridge domain names. If you do, the configuration does not commit and an error is generated.

For the `vlan-id` statement, you can specify either a valid VLAN identifier or the **none** or **all** options. For information about VLAN identifiers and VLAN tags for a bridge domain, see Configuring VLAN Identifiers for Bridge Domains and VPLS Routing Instances.

To include one or more logical interfaces in the bridge domain, specify an ***interface-name*** for an Ethernet interface you configured at the `[edit interfaces]` hierarchy level.

NOTE: A maximum of 4000 active logical interfaces are supported on a bridge domain or on each mesh group in a virtual private LAN service (VPLS) instance configured for Layer 2 bridging.

To configure a layer 2 logical interface to be included in a bridge domain, you can either include the `encapsulation vlan-bridge` statement under the logical interface, or the `encapsulation ethernet-bridge` statement under the physical interface.

NOTE: On ACX Series routers, a maximum of 1000 logical interfaces can be configured on a physical interface. You can configure a maximum of 3000 bridge domains on an ACX Series router.

By default, each bridge domain maintains a Layer 2 forwarding database that contains media access control (MAC) addresses learned from packets received on the ports that belong to the bridge domain. You can modify Layer 2 forwarding properties, including disabling MAC learning for the entire system or a bridge domain, adding static MAC addresses for specific logical interfaces, and limiting the number of MAC addresses learned by the entire system, the bridge domain, or a logical interface.

You can also configure spanning tree protocols to prevent forwarding loops. .

In Junos OS Release 8.5 and later, you can configure IGMP snooping for a bridge domain. For more information, see the Junos OS Multicast Protocols User Guide.

Integrated routing and bridging (IRB) provides simultaneous support for Layer 2 bridging and Layer 3 routing on the same interface. IRB enables you to route packets to another routed interface or to another bridge domain that has an IRB interface configured. You configure a logical routing interface by including the `irb` statement at the `[edit interfaces]` hierarchy level and include that interface in the bridge domain. For more information about how to configure a routing interface, see the Junos OS Network Interfaces Library for Routing Devices.

NOTE: You can include only one routing interface in a bridge domain.

To configure a bridge domain with IRB support, include the following statements:

```
[edit]
bridge-domains {
    bridge-domain-name {
        domain-type bridge;
        interface interface-name;
        routing-interface routing-interface-name;
        service-id number;
        vlan-id (none | number);
        vlan-tags outer number inner number;
    }
}
```

For each bridge domain that you configure, specify a ***bridge-domain-name***. You must also specify the value **bridge** for the `domain-type` statement.

For the `vlan-id` statement, you can specify either a valid VLAN identifier or the **none** option.

NOTE: If you configure a routing interface to support IRB in a bridge domain, you cannot use the **all** option for the `vlan-id` statement.

The `vlan-tags` statement enables you to specify a pair of VLAN identifiers; an **outer** tag and an **inner** tag.

NOTE: For a single bridge domain, you can include either the `vlan-id` statement or the `vlan-tags` statement, but not both.

For MC-LAG bridge domains, when the VLAN identifier is `none`, use the `service-id` statement to facilitate media access control (MAC) and Address Resolution Protocol (ARP) synchronization among MC-LAG peers.

To include one or more logical interfaces in the bridge domain, specify the interface name for each Ethernet interface to include that you configured at the `[edit interfaces]` hierarchy level.

NOTE: A maximum of 4000 active logical interfaces are supported on a bridge domain or on each mesh group in a VPLS routing instance configured for Layer 2 bridging.

To associate a routing interface with a bridge domain, include the `routing-interface` *`routing-interface-name`* statement and specify a ***routing-interface-name*** you configured at the `[edit interfaces irb]` hierarchy level. You can configure only one routing interface for each bridge domain. For more information about how to configure logical and routing interfaces, see the Junos OS Network Interfaces Library for Routing Devices.

In Junos OS Release 9.0 and later, IRB interfaces are supported for multicast snooping. For more information about multicast snooping, see the *Understanding Multicast Snooping and VPLS Root Protection*.

In Junos 11.4 and later, IP multicast is supported on Layer 2 trunk ports through IRB interfaces using the Trio chipset.

In Junos OS Release 9.6 and later, in multihomed VPLS configurations, you can configure VPLS to keep a VPLS connection up if only an IRB interface is available by configuring the **irb** option for the `connectivity-type` statement at the `[edit routing-instances` *`routing-instance-name`* `protocols vpls]` hierarchy level. The `connectivity-type` statement has two options, **ce** and **irb**. The **ce** option is the default and specifies that a CE interface is required to maintain the VPLS connection. By default, if only an IRB interface is available, the VPLS connection is brought down. For more information about configuring VPNs, see the *Junos VPN Configuration Guide*.

NOTE: When you configure IRB interfaces in more than one logical system on a device, all of the of the IRB logical interfaces share the same MAC address.

Integrated Bridging and Routing (IRB) interfaces are used to tie together Layer 2 switched and Layer 3 routed domains on MX routers. MX routers support classifiers and rewrite rules on the IRB interface at

the [edit class-of-service interfaces irb unit logical-unit-number] level of the hierarchy. All types of classifiers and rewrite rules are allowed, including IEEE 802.1p.

NOTE: The IRB classifiers and rewrite rules are used only for *routed* packets; in other words, it is for traffic that originated in the Layer 2 domain and is then routed through IRB into the Layer 3 domain, or vice versa. Only IEEE classifiers and IEEE rewrite rules are allowed for pure Layer 2 interfaces within a bridge domain.

Configuring VLAN Identifiers for Bridge Domains and VPLS Routing Instances

For a bridge domain that is performing Layer 2 switching only, you do not have to specify a VLAN identifier.

For a bridge domain that is performing Layer 3 IP routing, you must specify either a VLAN identifier or dual VLAN identifier tags.

For a VPLS routing instance, you must specify either a VLAN identifier or dual VLAN identifier tags.

You can configure VLAN identifiers for a bridge domain or a VPLS routing instance in the following ways:

- By using the **input-vlan-map** and the output-vlan-map statements at the [edit interfaces *interface-name*] or [edit logical-systems *logical-system-name* interfaces *interface-name*] hierarchy level to configure VLAN mapping. For information about configuring input and output VLAN maps to stack and rewrite VLAN tags in incoming or outgoing frames, see the Junos OS Network Interfaces Library for Routing Devices.
- By using either the vlan-id statement or the vlan-tags statement to configure a normalizing VLAN identifier. This topic describes how normalizing VLAN identifiers are processed and translated in a bridge domain or a VPLS routing instance.

The **vlan-id** and vlan-tags statements are used to specify the normalizing VLAN identifier under the bridge domain or VPLS routing instance. The normalizing VLAN identifier is used to perform the following functions:

- Translate, or normalize, the VLAN tags of received packets received into a learn VLAN identifier.
- Create multiple learning domains that each contain a learn VLAN identifier. A learning domain is a MAC address database to which MAC addresses are added based on the learn VLAN identifier.

NOTE: You cannot configure VLAN mapping using the **input-vlan-map** and `output-vlan-map` statements if you configure a normalizing VLAN identifier for a bridge domain or VPLS routing instance using the **vlan-id** or `vlan-tags` statements.

To configure a VLAN identifier for a bridge domain, include either the **vlan-id** or the `vlan-tags` statement at the `[edit interfaces` *`interface-name`* `unit` *`logic-unit-number`* `family bridge]` or `[edit logical-systems` *`logical-system-name`* `interfaces` *`interface-name`* `unit` *`logic-unit-number`* `family bridge]` hierarchy level, and then include that logical interface in the bridge domain configuration. For more information about configuring a bridge domain, see Configuring a Bridge Domain.

For a VPLS routing instance, include either the **vlan-id** or `vlan-tags` statement at the `[edit interfaces` *`interface-name`* `unit` *`logic-unit-number`*`]` or `[edit logical-systems` *`logical-system-name`* `interfaces` *`interface-name`* `unit` *`logic-unit-number`*`]` hierarchy level, and then include that logical interface in the VPLS routing instance configuration. For more information about configuring a VPLS routing instance, see the Junos OS VPNs Library for Routing Devices.

NOTE: The maximum number of Layer 2 interfaces that you can associate with a bridge domain or a VPLS instance on MX Series routers is 4000.

NOTE: For a single bridge domain or VPLS routing instance, you can include either the **vlan-id** or the `vlan-tags` statement, but not both. If you do not configure a **vlan-id**, **vlan-tags**, or **vlan-id-list [** ***vlan-id-numbers*** **]** for the bridge domain or the VPLS routing instance, the Layer 2 packets received are forwarded to the outbound Layer 2 interface without having the VLAN tag modified unless an **output-vlan-map** is configured on the Layer 2 interface. This results in a frame being forwarded to a Layer 2 interface with a VLAN tag that is different from what is configured for the Layer 2 interface. Note that a frame received from the Layer 2 interface is still required to match the VLAN tag(s) specified in the interface configuration. The invalid configuration may cause a Layer 2 loop to occur.

The VLAN tags associated with the inbound logical interface are compared with the normalizing VLAN identifier. If the tags are different, they are rewritten as described in Table 3 on page 15. The source MAC address of a received packet is learned based on the normalizing VLAN identifier.

NOTE: You do not have to specify a VLAN identifier for a bridge domain that is performing Layer 2 switching only. To support Layer 3 IP routing, you must specify either a VLAN identifier or a pair of VLAN tags. However, you cannot specify the same VLAN

identifier for more than one bridge domain within a routing instance. Each bridge domain must have a unique VLAN identifier.

If the VLAN tags associated with the outbound logical interface and the normalizing VLAN identifier are different, the normalizing VLAN identifier is rewritten to match the VLAN tags of the outbound logical interface, as described in Table 4 on page 16.

For the packets sent over the VPLS routing instance to be tagged by the normalizing VLAN identifier, include one of the following configuration statements:

- **vlan-id *number*** to tag all packets that are sent over the VPLS virtual tunnel (VT) interfaces with the VLAN identifier.
- **vlan-tags outer *number* inner *number*** to tag all packets sent over the VPLS VT interfaces with dual outer and inner VLAN tags.

Use the `vlan-id none` statement to have the VLAN tags removed from packets associated with an inbound logical interface when those packets are sent over VPLS VT interfaces. Note that those packets might still be sent with other customer VLAN tags.

The `vlan-id all` statement enables you to configure bridging for several VLANs with a minimum amount of configuration. Configuring this statement creates a learning domain for:

- Each inner VLAN, or learn VLAN, identifier of a logical interface configured with two VLAN tags
- Each VLAN, or learn VLAN, identifier of a logical interface configured with one VLAN tag

We recommend that you do not use customer VLAN IDs in a VPLS routing instance because customer VLAN IDs are used for learning only.

You should use the service VLAN ID in a VPLS routing instance, as in the following configuration:

```
[edit]
interface ge-1/1/1 {
    vlan-tagging;
    unit 1 {
        vlan-id s1; /* Service vlan */
        encapsulation vlan-vpls;
        input-vlan-map pop; /* Pop the service vlan on input */
        output-vlan-map push; /* Push the service vlan on output */
    }
}
interface ge-1/1/2 {
    encapsulation ethernet-vpls;
    unit 0;
```

```
}
routing-instance {
    V1 {
        instance-type vpls;
        vlan-id all;
        interface ge-1/1/1.1;
        interface ge-1/1/2.0;
    }
}
```

NOTE: If you configure the `vlan-id all` statement in a VPLS routing instance, we recommend using the **input-vlan-map pop** and `output-vlan-map push` statements on the logical interface to pop the service VLAN ID on input and push the service VLAN ID on output and in this way limit the impact of doubly-tagged frames on scaling. You cannot use the native `vlan- id` statement when the `vlan-id all` statement is included in the configuration.

The `vlan-id-list [ vlan-id-numbers ]` statement enables you to configure bridging for multiple VLANs on a trunk interface. Configuring this statement creates a learning domain for:

- Each VLAN listed: `vlan-id-list [ 100 200 300 ]`
- Each VLAN in a range: `vlan-id-list [ 100-200 ]`
- Each VLAN in a list and range combination: `vlan-id-list [ 50, 100-200, 300 ]`

The following steps outline the process for bridging a packet received over a Layer 2 logical interface when you specify a normalizing VLAN identifier using either the **vlan-id *number*** or `vlan-tags` statement for a bridge domain or a VPLS routing instance:

1. When a packet is received on a physical port, it is accepted only if the VLAN identifier of the packet matches the VLAN identifier of one of the logical interfaces configured on that port.
2. The VLAN tags of the received packet are then compared with the normalizing VLAN identifier. If the VLAN tags of the packet are different from the normalizing VLAN identifier, the VLAN tags are rewritten as described in Table 3 on page 15.
3. If the source MAC address of the received packet is not present in the source MAC table, it is learned based on the normalizing VLAN identifier.
4. The packet is then forwarded toward one or more outbound Layer 2 logical interfaces based on the destination MAC address. A packet with a known unicast destination MAC address is forwarded only to one outbound logical interface. For each outbound Layer 2 logical interface, the normalizing VLAN identifier configured for the bridge domain or VPLS routing instance is compared with the VLAN tags configured on that logical interface. If the VLAN tags associated with an outbound logical interface

do not match the normalizing VLAN identifier configured for the bridge domain or VPLS routing instance, the VLAN tags are rewritten as described in Table 4 on page 16.

The tables below show how VLAN tags are applied for traffic sent to and from the bridge domain, depending on how the **vlan-id** and `vlan-tags` statements are configured for the bridge domain and on how VLAN identifiers are configured for the logical interfaces in a bridge domain or VPLS routing instance. Depending on your configuration, the following rewrite operations are performed on VLAN tags:

- **pop**—Remove a VLAN tag from the top of the VLAN tag stack.
- **pop-pop**—Remove both the outer and inner VLAN tags of the frame.
- **pop-swap**—Remove the outer VLAN tag of the frame and replace the inner VLAN tag of the frame.
- **swap**—Replace the VLAN tag of the frame.
- **push**—Add a new VLAN tag to the top of the VLAN stack.
- **push-push**—Push two VLAN tags in front of the frame.
- **swap-push**—Replace the VLAN tag of the frame and add a new VLAN tag to the top of the VLAN stack.
- **swap-swap**—Replace both the outer and inner VLAN tags of the frame.

Table 3 on page 15 shows specific examples of how the VLAN tags for packets sent to the bridge domain are processed and translated, depending on your configuration. "–" means that the statement is not supported for the specified logical interface VLAN identifier. "No operation" means that the VLAN tags of the received packet are not translated for the specified input logical interface.

Table 3: Statement Usage and Input Rewrite Operations for VLAN Identifiers for a Bridge Domain

VLAN Identifier of Logical Interface	VLAN Configurations for Bridge Domain			
	vlan-id none	vlan-id 200	vlan-id all	vlan tags outer 100 inner 300
none	No operation	push 200	–	push 100, push 300
200	pop 200	No operation	No operation	swap 200 to 300, push 100
1000	pop 1000	swap 1000 to 200	No operation	swap 1000 to 300, push 100

Table 3: Statement Usage and Input Rewrite Operations for VLAN Identifiers for a Bridge Domain *(Continued)*

VLAN Identifier of Logical Interface	VLAN Configurations for Bridge Domain			
	vlan-id none	vlan-id 200	vlan-id all	vlan tags outer 100 inner 300
vlan-tags outer 2000 inner 300	pop 2000, pop 300	pop 2000, swap 300 to 200	pop 2000	swap 2000 to 100
vlan-tags outer 100 inner 400	pop 100, pop 400	pop 100, swap 400 to 200	pop 100	swap 400 to 300
vlan-id-range 10-100	–	–	No operation	–
vlan-tags outer 200 inner-range 10-100	–	–	pop 200	–

Table 4 on page 16 shows specific examples of how the VLAN tags for packets sent from the bridge domain are processed and translated, depending on your configuration. "–" means that the statement is not supported for the specified logical interface VLAN identifier. "No operation" means that the VLAN tags of the outbound packet are not translated for the specified output logical interface.

Table 4: Statement Usage and Output Rewrite Operations for VLAN Identifiers for a Bridge Domain

VLAN Identifier of Logical Interface	VLAN Configurations for Bridge Domain			
	vlan-id none	vlan-id 200	vlan-id all	vlan tags outer 100 inner 300
none	no operation	pop 200	–	pop 100, pop 300
200	push 200	No operation	No operation	pop 100, swap 300 to 200

Table 4: Statement Usage and Output Rewrite Operations for VLAN Identifiers for a Bridge Domain *(Continued)*

VLAN Identifier of Logical Interface	VLAN Configurations for Bridge Domain			
	vlan-id none	vlan-id 200	vlan-id all	vlan tags outer 100 inner 300
1000	push 1000	swap 200 to 1000	No operation	pop 100, swap 300 to 1000
vlan-tags outer 2000 inner 300	push 2000, push 300	swap 200 to 300, push 2000	push 2000	swap 100 to 2000
vlan-tags outer 100 inner 400	push 100, push 400	swap 200 to 400, push 100	push 100	swap 300 to 400
vlan-id-range 10-100	–	–	No operation	–
vlan-tags outer 200 inner-range 10-100	–	–	push 200	–

Configuring VLAN Identifiers for Bridge Domains in ACX Series

You can configure VLAN identifiers for a bridge domain for normalization in the following ways:

- Configure VLAN mapping by using the **input-vlan-map** and the `output-vlan-map` statements at the `[edit interfaces interface-name]` hierarchy level.
- Configure an implicit normalizing VLAN identifier under the bridge domain by using the **vlan-id** statement at the `[edit bridge-domains bridge-domain-name]` hierarchy level.

NOTE: You cannot configure VLAN mapping by using the **input-vlan-map** and `output-vlan-map` statements if you configure a normalizing VLAN identifier for a bridge domain by using the **vlan-id** statement.

You can use the `vlan-id-list [ vlan-id-numbers ]` statement to configure bridging for multiple VLANs. Configuring this statement creates a bridge domain for:

- Each VLAN listed—for example, `vlan-id-list [ 100 200 300 ]`
- Each VLAN in a range—for example, `vlan-id-list [ 100-200 ]`
- Each VLAN in a list and range combination—for example, `vlan-id-list [ 50, 100-200, 300 ]`

Example: Configuring Basic Layer 2 Switching on MX Series

IN THIS SECTION

This example shows how to configure Layer 2 switching with all interfaces participating in a single VLAN.

Requirements

No special configuration beyond device initialization is required before configuring this example.

This example uses an MX Series device to perform Layer 2 switching.

Overview

IN THIS SECTION

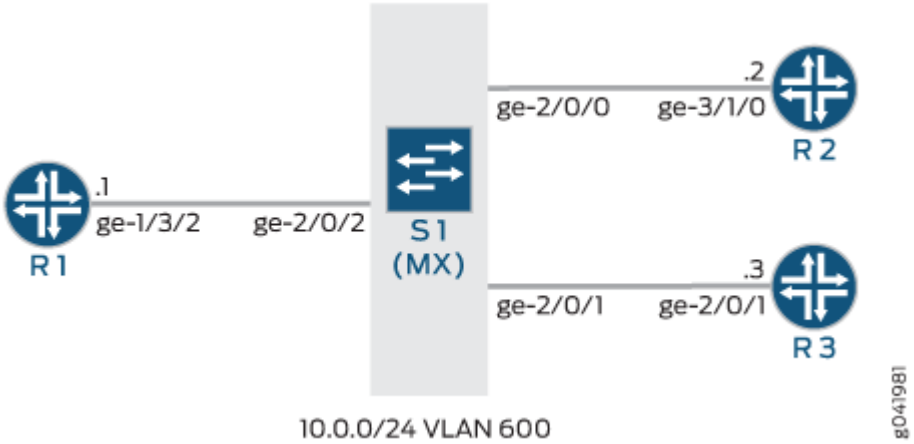
In this example, a single MX Series device is configured to act as a basic single-VLAN switch. Three connections are in place. The connections from the MX Series device attach to Junos OS routers, but

the routers are used here for testing purposes only. In place of routers, you can use any IP networking devices.

Topology

Figure 1 on page 19 shows the sample network.

Figure 1: Basic Layer 2 Switching



"CLI Quick Configuration" on page 19 shows the configuration for all of the devices in Figure 1 on page 19.

Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

Device S1

```
set interfaces ge-2/0/0 vlan-tagging
set interfaces ge-2/0/0 encapsulation extended-vlan-bridge
set interfaces ge-2/0/0 unit 0 vlan-id 600
set interfaces ge-2/0/1 vlan-tagging
set interfaces ge-2/0/1 encapsulation extended-vlan-bridge
set interfaces ge-2/0/1 unit 0 vlan-id 600
set interfaces ge-2/0/2 vlan-tagging
set interfaces ge-2/0/2 encapsulation extended-vlan-bridge
set interfaces ge-2/0/2 unit 0 vlan-id 600
set bridge-domains customer1 domain-type bridge
set bridge-domains customer1 interface ge-2/0/0.0
set bridge-domains customer1 interface ge-2/0/2.0
set bridge-domains customer1 interface ge-2/0/1.0
```

Device R1

```
set interfaces ge-1/3/2 vlan-tagging
set interfaces ge-1/3/2 unit 0 vlan-id 600
set interfaces ge-1/3/2 unit 0 family inet address 10.0.0.1/24
```

Device R2

```
set interfaces ge-3/1/0 vlan-tagging
set interfaces ge-3/1/0 unit 0 vlan-id 600
set interfaces ge-3/1/0 unit 0 family inet address 10.0.0.2/24
```

Device R3

```
set interfaces ge-2/0/1 vlan-tagging
set interfaces ge-2/0/1 unit 0 vlan-id 600
set interfaces ge-2/0/1 unit 0 family inet address 10.0.0.3/24
```

Procedure

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode* in the Junos OS CLI User Guide.

To configure Device S1:

1. Configure the device interfaces.

```
[edit interfaces]
user@S1# set interfaces ge-2/0/0 vlan-tagging
user@S1# set interfaces ge-2/0/0 encapsulation extended-vlan-bridge
user@S1# set interfaces ge-2/0/0 unit 0 vlan-id 600
user@S1# set interfaces ge-2/0/1 vlan-tagging
user@S1# set interfaces ge-2/0/1 encapsulation extended-vlan-bridge
user@S1# set interfaces ge-2/0/1 unit 0 vlan-id 600
user@S1# set interfaces ge-2/0/2 vlan-tagging
user@S1# set interfaces ge-2/0/2 encapsulation extended-vlan-bridge
user@S1# set interfaces ge-2/0/2 unit 0 vlan-id 600
```

2. Configure the bridge domain.

```
[edit interfaces]
user@S1# set bridge-domains customer1 domain-type bridge
user@S1# set bridge-domains customer1 interface ge-2/0/0.0
user@S1# set bridge-domains customer1 interface ge-2/0/2.0
user@S1# set bridge-domains customer1 interface ge-2/0/1.0
```

Results

From configuration mode, confirm your configuration by entering the `show interfaces` and `show bridge-domains` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@S1# show interfaces
ge-2/0/0 {
    vlan-tagging;
```

```
    encapsulation extended-vlan-bridge;
    unit 0 {
        vlan-id 600;
    }
}
ge-2/0/1 {
    vlan-tagging;
    encapsulation extended-vlan-bridge;
    unit 0 {
        vlan-id 600;
    }
}
ge-2/0/2 {
    vlan-tagging;
    encapsulation extended-vlan-bridge;
    unit 0 {
        vlan-id 600;
    }
}
```

```
user@S1# show bridge-domains
customer1 {
    domain-type bridge;
    interface ge-2/0/0.0;
    interface ge-2/0/2.0;
    interface ge-2/0/1.0;
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Confirming the MAC Address Learning

Purpose

Display Layer 2 MAC address information.

Action

- From Device S1, run the `show bridge mac-table` command.

```
user@S1> show bridge mac-table

MAC flags (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
           SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

Routing instance : default-switch
 Bridging domain : customer1, VLAN : NA
   MAC                MAC      Logical          NH     RTR
   address            flags    interface        Index  ID
   00:12:1e:ee:34:dd  D        ge-2/0/2.0
   00:1d:b5:5e:86:79  D        ge-2/0/0.0
   00:21:59:0f:35:2b  D        ge-2/0/1.0
```

- From Device S1, run the `show bridge mac-table extensive` command.

```
user@S1> show bridge mac-table extensive

MAC address: 00:12:1e:ee:34:dd
  Routing instance: default-switch
   Bridging domain: customer1, VLAN : NA
   Learning interface: ge-2/0/2.0
   Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
```

```
    Epoch: 1                          Sequence number: 0
    Learning mask: 0x00000004

MAC address: 00:1d:b5:5e:86:79
  Routing instance: default-switch
    Bridging domain: customer1, VLAN : NA
    Learning interface: ge-2/0/0.0
    Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
    Epoch: 1                          Sequence number: 0
    Learning mask: 0x00000004

MAC address: 00:21:59:0f:35:2b
  Routing instance: default-switch
    Bridging domain: customer1, VLAN : NA
    Learning interface: ge-2/0/1.0
    Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
    Epoch: 3                          Sequence number: 0
    Learning mask: 0x00000004
```

Meaning

The output shows that the MAC addresses have been learned.

Making Sure That the Attached Devices Can Reach Each Other

Purpose

Verify connectivity.

Action

```
user@R1> ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2): 56 data bytes
64 bytes from 10.0.0.2: icmp_seq=0 ttl=64 time=1.178 ms
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.192 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.149 ms
^C
--- 10.0.0.2 ping statistics ---
```

```
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.149/1.173/1.192/0.018 ms
```

```
user@R1> ping 10.0.0.3
PING 10.0.0.3 (10.0.0.3): 56 data bytes
64 bytes from 10.0.0.3: icmp_seq=0 ttl=64 time=1.189 ms
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=1.175 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=1.178 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=1.133 ms
^C
--- 10.0.0.3 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.133/1.169/1.189/0.021 ms
```

```
user@R2> ping 10.0.0.3
PING 10.0.0.3 (10.0.0.3): 56 data bytes
64 bytes from 10.0.0.3: icmp_seq=0 ttl=64 time=0.762 ms
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.651 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.722 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.705 ms
^C
--- 10.0.0.3 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.651/0.710/0.762/0.040 ms
```

Meaning

The output shows that the attached devices have established Layer 3 connectivity, with Device S1 doing transparent Layer 2 bridging.

Checking the Bridge Domain

Purpose

Display bridge domain information.

Action

```
user@S1> show bridge domain extensive

Routing instance: default-switch
Bridge domain: customer1                      State: Active
Bridge VLAN ID: NA
Interfaces:
    ge-2/0/0.0
    ge-2/0/1.0
    ge-2/0/2.0
Total MAC count: 3
```

Meaning

The output shows that bridge domain is active.

Checking the Bridge Statistics

Purpose

Display bridge statistics.

Action

```
user@S1> show bridge statistics


   Local interface: ge-2/0/0.0, Index: 65543
     Broadcast packets:                     0
     Broadcast bytes  :                     0
     Multicast packets:                    80
     Multicast bytes  :                  8160
     Flooded packets  :                     0
     Flooded bytes    :                     0
     Unicast packets  :                     1
     Unicast bytes    :                    64
     Current MAC count:                     1 (Limit 1024)
   Local interface: ge-2/0/2.0, Index: 324
     Broadcast packets:                     0
```

```
    Broadcast bytes  :                     0
    Multicast packets:                    80
    Multicast bytes  :                  8160
    Flooded packets  :                     1
    Flooded bytes    :                    74
    Unicast packets  :                    52
    Unicast bytes    :                  4332
    Current MAC count:                     1 (Limit 1024)
  Local interface: ge-2/0/1.0, Index: 196613
    Broadcast packets:                     2
    Broadcast bytes  :                   128
    Multicast packets:                     0
    Multicast bytes  :                     0
    Flooded packets  :                     1
    Flooded bytes    :                    93
    Unicast packets  :                    51
    Unicast bytes    :                  4249
    Current MAC count:                     1 (Limit 1024)
```

Meaning

The output shows that bridge domain interfaces are sending and receiving packets.

Checking the Bridge Flooding

Purpose

Display bridge flooding information.

Action

```
user@S1> show bridge flood extensive

Name: __juniper_private1__
CEs: 0
VEs: 0
Name: default-switch
CEs: 3
VEs: 0
Bridging domain: customer1
  Flood route prefix: 0x30003/51
```

```
Flood route type: FLOOD_GRP_COMP_NH
Flood route owner: __all_ces__
Flood group name: __all_ces__
Flood group index: 1
Nexthop type: comp
Nexthop index: 568
  Flooding to:
  Name             Type          NhType          Index
  __all_ces__      Group          comp            562
      Composition: split-horizon
      Flooding to:
      Name             Type          NhType          Index
      ge-2/0/0.0       CE             ucst            524
      ge-2/0/1.0       CE             ucst            513
      ge-2/0/2.0       CE             ucst            523

Flood route prefix: 0x30005/51
Flood route type: FLOOD_GRP_COMP_NH
Flood route owner: __re_flood__
Flood group name: __re_flood__
Flood group index: 65534
Nexthop type: comp
Nexthop index: 565
  Flooding to:
  Name             Type          NhType          Index
  __all_ces__      Group          comp            562
      Composition: split-horizon
      Flooding to:
      Name             Type          NhType          Index
      ge-2/0/0.0       CE             ucst            524
      ge-2/0/1.0       CE             ucst            513
      ge-2/0/2.0       CE             ucst            523
```

Meaning

If the destination MAC address of a packet is unknown to the device (that is, the destination MAC address in the packet does not have an entry in the forwarding table), the device duplicates the packet and floods it on all interfaces in the bridge domain other than the interface on which the packet arrived. This is known as packet flooding and is the default behavior for the device to determine the outgoing interface for an unknown destination MAC address.

Checking Layer 2 Learning

Purpose

Display Layer 2 learning information for all the interfaces.

Action

```
user@S1> show l2-learning interface

Routing Instance Name : default-switch
Logical Interface flags (DL -disable learning, AD -packet action drop,
                         LH - MAC limit hit, DN - Interface Down )
Logical             BD         MAC        STP          Logical
Interface           Name       Limit      State        Interface flags
ge-2/0/2.0                     0
                    custom..   1024       Forwarding
Routing Instance Name : default-switch
Logical Interface flags (DL -disable learning, AD -packet action drop,
                         LH - MAC limit hit, DN - Interface Down )
Logical             BD         MAC        STP          Logical
Interface           Name       Limit      State        Interface flags
ge-2/0/0.0                     0
                    custom..   1024       Forwarding
Routing Instance Name : default-switch
Logical Interface flags (DL -disable learning, AD -packet action drop,
                         LH - MAC limit hit, DN - Interface Down )
Logical             BD         MAC        STP          Logical
Interface           Name       Limit      State        Interface flags
ge-2/0/1.0                     0
                    custom..   1024       Forwarding
```

2

CHAPTER

Layer 2 Address Learning and Forwarding

Layer 2 Address Learning and Forwarding Overview

SUMMARY

IN THIS SECTION

Understanding Layer 2 Learning and Forwarding

On MX Series routers only, you can configure Layer 2 MAC address and VLAN learning and forwarding properties in support of Layer 2 bridging. The router learns unicast media access control (MAC) addresses to avoid flooding the packets to all the ports in a bridge domain. The MX Series router creates a source MAC entry in its source and destination MAC tables for each MAC address learned from packets received on ports that belong to the bridge domain. If the bridge domain receives a control protocol data unit (PDU) which does not have a corresponding protocol configured, then the control PDU is considered as an unknown multicast data packet and the packets are flooded across all the ports that are part of the same bridge domain. If the bridge domain has the protocol corresponding to the PDU configured , then the control PDU is considered as a control packet and is processed by the routing engine.

By default, Layer 2 address learning is enabled. You can disable MAC learning for the router or for a specific bridge domain or logical interfaces. You can also configure the following Layer 2 forwarding properties for an MX Series router:

- Timeout interval for MAC entries
- MAC accounting
- A limit to the number of MAC addresses learned from the logical interfaces

Understanding Layer 2 Learning and Forwarding for Bridge Domains

When you configure a bridge domain, Layer 2 address learning is enabled by default. The bridge domain learns unicast media access control (MAC) addresses to avoid flooding the packets to all the ports in the bridge domain. Each bridge domain creates a source MAC entry in its source and destination MAC tables for each source MAC address learned from packets received on the ports that belong to the bridge domain.

NOTE: Traffic is not flooded back onto the interface on which it was received. However, because this "split horizon" occurs at a late stage, the packet statistics displayed by commands such as `show interfaces queue` will include flood traffic.

You can optionally disable MAC learning either for the entire router or for a specific bridge domain or *logical interface*. You can also configure the following Layer 2 learning and forwarding properties:

- Static MAC entries for logical interfaces only
- Limit to the number of MAC addresses learned from a specific logical interface or from all the logical interfaces in a bridge domain
- Size of the MAC address table for the bridge domain
- MAC accounting for a bridge domain

Configure MAC Address for Layer 2 Learning and Forwarding

SUMMARY

IN THIS SECTION

Configuring Static MAC Addresses for Logical Interfaces in a Bridge Domain

You can manually add static MAC entries for the logical interfaces in a bridge domain. You can specify one or more static MAC addresses for each logical interface.

To add a static MAC address for a logical interface in a bridge domain, include the `static-mac` *`mac-address`* statement at the `[edit bridge-domains` *`bridge-domain-name`* `bridge-options interface` *`interface-name`*`]` hierarchy level.

```
[edit]
bridge-domains {
    bridge-domain-name {
        domain-type bridge;
        bridge-options {
            interface interface-name {
                static-mac mac-address {
                    <vlan-id number>;
                }
            }
        }
```

```
    }
}
```

You can optionally specify a VLAN identifier for the static MAC address by using the `vlan-id` statement. To specify a VLAN identifier for a static MAC address, you must use the **all** option when configuring a VLAN identifier for the bridge domain.

NOTE: If a static MAC address you configure for a logical interface appears on a different logical interface, packets sent to that interface are dropped.

Configuring the Size of the MAC Address Table for a Bridge Domain

You can modify the size of the MAC address table for each bridge domain. The default table size is 5120 addresses. The minimum you can configure is 16 addresses, and the maximum is 1,048,575 addresses.

If the MAC table limit is reached, new addresses can no longer be added to the table. Unused MAC addresses are removed from the MAC address table automatically. This frees space in the table, allowing new entries to be added.

To modify the size of the MAC table, include the `mac-table-size` *`limit`* statement at the `[edit bridge-domains` *`bridge-domain-name`* `bridge-options]` hierarchy level:

```
[edit]
bridge-domains {
    bridge-domain-name {
        domain-type bridge;
        bridge-options {
            mac-table-size limit {
                packet-action drop;
            }
        }
    }
}
```

Limiting MAC Addresses Learned from an Interface in a Bridge Domain

You can configure a limit on the number of MAC addresses learned from a specific bridge domain or from a specific logical interface that belongs to a bridge domain.

To configure a limit for the number of MAC addresses learned from each logical interface in a bridge domain, include the `interface-mac-limit` *`limit`* statement at the `[edit bridge-domains` *`bridge-domain-name`* `bridge-options]` hierarchy level:

```
[edit]
bridge-domains {
    bridge-domain-name {
        domain-type bridge;
        interface interface-name;
        bridge-options {
            interface-mac-limit limit;
        }
    }
}
```

To limit the number of MAC addresses learned from a specific logical interface in a bridge domain or an entire bridge domain, include the `interface-mac-limit` *`limit`* statement at the `[edit bridge-domains` *`bridge-domain-name`* `bridge-options interface` *`interface-name`*`]` or `[edit bridge-domains` *`bridge-domain-name`* `bridge-options]` hierarchy level:

```
[edit]
bridge-domains {
    bridge-domain-name {
        domain-type bridge;
        interface interface-name;
        bridge-options {
            interface-mac-limit limit{
                packet-action drop;
            }
            interface interface-name {
                interface-mac-limit limit{
                    packet-action drop;
                }
            }
        }
```

```
    }
}
```

For an access port, the default limit on the maximum number of MAC addresses that can be learned on an access port is 1024. Because an access port can be configured in only one bridge domain in a network topology, the default limit is 1024 addresses, which is same as the limit for MAC addresses learned on a logical interface in a bridge domain (configured by including the `interface-mac-limit` *`limit`* statement at the `[edit bridge-domains` *`bridge-domain-name`* `bridge-options interface` *`interface-name`*`]` or `[edit bridge-domains` *`bridge-domain-name`* `bridge-options]` hierarchy level.

For a trunk port, the default limit on the maximum number of MAC addresses that can be learned on a trunk port is 8192. Because a trunk port can be associated with multiple bride domains, the default limit is the same as the limit for MAC addresses learned on a logical interface in a virtual switch instance (configured by including the `interface-mac-limit` *`limit`* statement at the `[edit routing-instances` *`routing-instance-name`* `switch- options interface` *`interface-name`*`]` for a virtual switch instance).

The value you configure for a specific logical interface overrides any value you specify for the entire bridge domain at the `[edit bridge-domains` *`bridge-domain-name`* `bridge-options]` hierarchy level.

The default limit to the number of MAC addresses that can be learned on a logical interface is 1024. The range that you can configure for a specific logical interface is 1 through 131,071.

After the MAC address limit is reached, the default is for any incoming packets with a new source MAC address to be forwarded. You can specify that the packets be dropped by including the `packet-action drop` statement. To specify that packets be dropped for the entire bridge domain, include the `packet-action drop` statement at the `[edit bridge-domains` *`bridge-domain-name`* `bridge-options interface-mac-limit` *`limit`*`]` hierarchy level:

```
[edit bridge-domains bridge-domain-name bridge-options interface-mac-limit limit]
packet-action drop;
```

To specify that the packets be dropped for a specific logical interface in a bridge domain, include the `packet-action drop` statement at the `[edit bridge-domains` *`bridge-domain-name`* `bridge-options interface` *`interface-name`* `interface-mac-limit` *`limit`*`]` hierarchy level:

```
[edit bridge-domains bridge-domain-name bridge-options interface interface-name interface-mac-limit
limit]
packet-action drop;
```

NOTE: The behavior is different for some configurations. For aggregated Ethernet interfaces and label-switched interfaces, the behavior is to learn all the new MAC addresses even when the limit has been reached. The excess addresses are later deleted. The learning limit does not apply to bridge domain trunk ports, because they have no counters for the individual domains, and those domains might have different MAC learning limits.

NOTE: When static MAC addresses are configured, the learning limit is the configured limit minus the number of static addresses.

NOTE: On MX Series routers running Junos OS Release 8.4 and later, statistics for an aged destination MAC entry are not retained. In addition, source and destination statistics are reset during a MAC move. In previous releases, only source statistics were reset during a MAC move.

You can also configure a limit to the number of MAC addresses learned for an MX Series router.

Configuring MAC Address Limits on a Logical Interface

IN THIS SECTION

You can configure a limit on the number of MAC addresses learned from a specific logical interface. This feature allows the MAC address table space to be distributed among different logical interfaces, thereby avoiding congestion. The MAC address limit can be applied for both VLAN and VPLS routing instances and by default the MAC limit depends on the profile configured. You can limit the number of MAC addresses learned for a bridge domain and a logical interface at the same time.

Configuring MAC Address Limit

You can configure the MAC Address limit by using the `set protocols l2-learning global-no-hw-mac-learning` CLI command.

NOTE: On ACX Series routers, MAC address limiting is supported only on ACX5000 line of routers.

The following configuration example enables limiting MAC address learning on logical interfaces:

```
[edit protocols]
l2-learning {
    global-no-hw-mac-learning;
}
```

You can configure a limit to the number of MAC addresses learned from the logical interfaces on an MX Series router.

To configure a limit to the total number of MAC addresses that can be learned from the logical interfaces, include the `global-mac-limit` *`limit`* statement at the [`edit protocols l2-learning`] hierarchy level:

The default limit to the number of MAC addresses that can be learned the router as a whole is 393,215. The range that you can configure for the router as a whole is 20 through 1,048,575.

After the configured MAC address limit is reached, the default is for packets to be forwarded. You can specify that the packets be dropped by including the `packet-action drop` statement at the [`edit protocols l2-learning global-mac-limit`] hierarchy level:

```
[edit]
protocols {
    l2-learning {
        global-mac-limit limit {
            packet-action drop;
        }
    }
}
```

Configuring MAC Address Limit for VLANs

To configure a limit for the number of MAC addresses learned on each logical interface in a VLAN, include the `interface-mac-limit` *`limit`* statement at the [`edit vlans` *`vlan-name`*] hierarchy level. To limit the

MAC addresses learned on a specific logical interface of the VLAN, include the `interface-mac-limit` *`limit`* statement at the [`edit vlans` *`vlan-name`* `interface` *`interface-name`*] hierarchy level. To limit the MAC addresses learned on each of the logical interfaces of the VLAN, include the `interface-mac-limit` *`limit`* statement at the [`edit vlans` *`vlan-name`* `switch-options`] hierarchy level.

The following example configures a limit for the number of MAC addresses learned on a logical interface in a VLAN:

```
[edit vlans]
vlan10 {
    interface ge-0/0/3.1;
    interface ge-0/0/1.5;
    switch-options {
        interface-mac-limit {
            10;
        }
    }
    interface ge-0/0/1.5 {
        interface-mac-limit {
            20;
        }
    }
}
```

Configuring MAC Address Limit for VPLS

To configure a limit for the number of MAC addresses learned on each logical interface in a VPLS routing instance, include the `interface-mac-limit` *`limit`* statement at the [`edit routing-instances` *`routing-instance-name`* `protocols vpls`] hierarchy level. To limit the MAC addresses learned on a specific logical interface of the VPLS instance, include the `interface-mac-limit` *`limit`* statement at the [`edit routing-instances` *`routing-instance-name`* `protocols vpls interface` *`interface-name`*] hierarchy level.

The following is an example to configure a limit for the number of MAC addresses learned on a logical interface in VPLS routing instance:

```
[edit routing-instance]
v1 {
    protocols {
        vpls {
            interface-mac-limit {
                10;
```

```
            }
            interface ge-0/0/1.3 {
                interface-mac-limit {
                    20;
                }
            }
        }
    }
}
```

If you have configured an interface MAC address limit for the logical interface in a bridge domain and a global MAC address limit for a bridge domain, then the interface MAC address limit is considered. The following example shows two MAC address limits configured on the interface ge-0/0/3.5 with the global value as 50 and local value as 30. In this case, the MAC address limit of 30 is considered for the interface ge-0/0/3.5 in the bridge domain.

```
vlan20 {
    interface ge-0/0/1.5;
    interface ge-0/0/3.5;
    switch-options {
        interface-mac-limit {
            50;
        }
        interface ge-0/0/1.5;
        interface ge-0/0/3.5 {
            interface-mac-limit {
                30;
            }
        }
    }
}
```

CLI Commands to Configure MAC Address Limiting

The following CLI commands are used for configuring MAC address limiting:

- `set protocols l2-learning global-no-hw-mac-learning`—Command to change the hardware-based MAC learning to software-based MAC learning mode.
- `set vlans` *`vlan-name`* `switch-options interface-mac-limit` *`limit`*—Command to configure the MAC address limit for each logical interface in a VLAN. The limit is applied to all logical interfaces belonging to the VLAN for which a separate interface MAC address limit is not configured.

- `set vlans vlan-name switch-options interface interface-name interface-mac-limit limit`—Command to configure the interface MAC address limit for a logical interface in a VLAN. The limit is applied to a specific logical interface in the VLAN for which it is configured.
- `set routing-instances routing-instance-name protocols vpls interface-mac-limit limit`—Command to configure the MAC address limit for each logical interface in the VPLS routing instance. This limit is applied to all logical interfaces belonging to the VPLS for which a separate interface MAC address limit is not configured.
- `set routing-instances routing-instance-name protocols vpls interface interface-name interface-mac-limit limit`—Command to configure the interface MAC address limit for a logical interface in the VPLS. This limit is applied to a specific logical interface in the VPLS for which it is configured.

Enabling MAC Accounting for a Router or a Bridge Domain

By default, MAC accounting is disabled. On MX Series routers, you can enable packet accounting either for the router as a whole or for a specific bridge domain. After you enable packet accounting, the Junos OS maintains packet counters for each MAC address learned.

To enable MAC accounting for an MX Series router, include the `global-mac-statistics` statement at the `[edit protocols l2-learning]` hierarchy level:

```
[edit protocols l2-learning]
global-mac-statistics;
```

To enable MAC accounting for a bridge domain, include the `mac-statistics` statement at the `[edit bridge-domains bridge-domain-name bridge-options]` hierarchy level:

```
[edit bridge-domains bridge-domain-name bridge-options]
mac-statistics;
```

Disabling MAC Learning for a Bridge Domain or Logical Interface

You can disable MAC learning for all logical interfaces in a specified bridge domain, or for a specific logical interface in a bridge domain. Disabling dynamic MAC learning prevents the specified interfaces from learning source MAC addresses.

To disable MAC learning for all logical interfaces in a bridge domain in a virtual switch, include the `no-mac-learning` statement at the `[edit bridge-domains *bridge-domain-name* bridge-options]` hierarchy level:

```
[edit]
bridge-domains {
    bridge-domain-name {
        domain-type bridge;
        interface interface-name;
        bridge-options {
            no-mac-learning;
        }
    }
}
```

To disable MAC learning for a specific logical interface in a bridge domain, include the `no-mac-learning` statement at the `[edit bridge-domains *bridge-domain-name* bridge-options interface *interface-name*]` hierarchy level.

```
[edit]
bridge-domains {
    bridge-domain-name {
        domain-type bridge;
        interface interface-name;
        bridge-options {
            interface interface-name {
                no-mac-learning;
            }
        }
    }
}
```

NOTE: When you disable MAC learning, source MAC addresses are not dynamically learned, and any packets sent to these source addresses are flooded into the bridge domain.

NOTE: When you gather interfaces into a bridge domain, the `no-mac-learn-enable` statement at the `[edit interfaces *interface-name* gigether-options ethernet-switch-profile]` hierarchy level is not supported. You must use the `no-mac-learning` statement at the `[edit`

`bridge-domains` *`bridge-domain-name`* `bridge-options interface` *`interface-name`*] hierarchy level to disable MAC learning on an interface in a bridge domain.

NOTE: When MAC learning is disabled for a VPLS routing instance, traffic is not load balanced and only one of the equal-cost next hops is used.

Configuring the MAC Table Timeout Interval

The MAC table aging process ensures that a router tracks only active MAC addresses on the network and is able to flush out address that are no longer used.

You can configure the MAC table aging time, the maximum time that an entry can remain in the MAC table before it "ages out," on all bridge domains, one or all VPLS instances, or one or all Ethernet virtual private network (EVPNs) instances on the router. This configuration can influence efficiency of network resource use by affecting the amount of traffic that is flooded to all interfaces because when traffic is received for MAC addresses no longer in the Ethernet routing table, the router floods the traffic to all interfaces.

Depending on how long you want to keep a MAC address in a MAC table before it expires, you can either increase or decrease the aging timer. By default, the timeout interval for all entries in the MAC table is 300 seconds. You can modify the timeout interval for MAC table entries on an MX Series router. You cannot modify the timeout interval for a virtual switch.

NOTE: The timeout interval applies only to dynamically learned MAC addresses. This value does not apply to configured static MAC addresses, which never time out.

The range for ***seconds*** is from 10 through 1,000,000.

You can modify the timeout interval for a router(at the global level) or on a per-domain basis (bridge domain).

- To modify the timeout interval for the MAC table for a router:

```
[edit protocols l2-learning]
user@host# set global-mac-table-aging-time time;
```

- To modify the timeout interval for a bridge domain:

```
[edit bridge-domain bridge-domain-name bridge-options];
user@host# set mac-table-aging-time time;
```

- To modify the timeout for a VPLS or an Ethernet virtual private network (EVPN) instance within a bridge domain:

```
[edit routing-instance routing-instance-name protocols vpls];
[edit routing-instance routing-instance-name protocols evpn];
user@host# set mac-table-aging-time time;
```

Example: Loop Detection Using the MAC Move Approach

IN THIS SECTION

This example shows how to detect loops using the MAC move approach.

Requirements

This example requires the following hardware and software components:

- MX Series 3D Universal Edge Routers
- Junos OS Release 13.2 running on all the devices

Overview

When a MAC address appears on a different physical interface or within a different unit of the same physical interface and if this behavior occurs frequently, it is considered a MAC move.

Configuration errors at the network can force traffic into never ending circular paths. Once there are loops in the Layer 2 network, one of the symptoms is frequent MAC moves, which can be used for rectification of the problem. When it is observed that a source MAC address is moving among the ports, interface is blocked based on the configured `action-priority` for the interface. If the `action-priority` value configured for interfaces is the same, the last interface for the bridge domain on which the MAC address move occurred is blocked.

Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level.

```
set interfaces ge-1/0/4 vlan-tagging
set interfaces ge-1/0/4 encapsulation flexible-ethernet-services
set interfaces ge-1/0/4 unit 10 encapsulation vlan-bridge
set interfaces ge-1/0/4 unit 10 vlan-id 10
set interfaces ge-1/0/4 unit 11 encapsulation vlan-bridge
set interfaces ge-1/0/4 unit 11 vlan-id 11
set interfaces ge-1/0/5 unit 0 family bridge interface-mode trunk
set interfaces ge-1/0/5 unit 0 family bridge vlan-id-list 10-12
set interfaces ge-1/0/6 unit 0 family bridge interface-mode trunk
set interfaces ge-1/0/6 unit 0 family bridge vlan-id-list 10-12
set bridge-domains bd10 vlan-id 10
set bridge-domains bd10 enable-mac-move-action
set bridge-domains bd10 bridge-options interface ge-1/0/5.0 action-priority 1
set bridge-domains bd10 bridge-options interface ge-1/0/6.0 action-priority 5
set bridge-domains bd11 vlan-id 11
set bridge-domains bd11 enable-mac-move-action
set bridge-domains bd12 vlan-id 12
```

In the previous example, all the interfaces, including the trunk interfaces in bd10 and bd11 will be monitored. If there are frequent MAC moves detected within interfaces ge-1/0/5 and ge-1/0/6, interface ge-1/0/5 is blocked. The blocking for trunk interfaces is such that data traffic only for a VLAN (on which the MAC move is detected) will be blocked and not for all the VLANs in the trunk. No action will be taken if a frequent MAC move is observed in **bd12**.

Configuring Loop Detection Using the MAC Move Approach

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see *Using the CLI Editor in Configuration Mode*.

To configure loop detection using the MAC address move approach:

1. Configure the interfaces.

```
[edit interfaces]
user@host# set ge-1/0/4 vlan-tagging
user@host# set ge-1/0/4 encapsulation flexible-ethernet-services
user@host# set ge-1/0/4 unit 10 encapsulation vlan-bridge
user@host# set ge-1/0/4 unit 10 vlan-id 10
user@host# set ge-1/0/4 unit 11 encapsulation vlan-bridge
user@host# set ge-1/0/4 unit 11 vlan-id 11
user@host# set ge-1/0/5 unit 0 family bridge interface-mode trunk
user@host# set ge-1/0/5 unit 0 family bridge vlan-id-list 10-12
user@host# set ge-1/0/6 unit 0 family bridge interface-mode trunk
user@host# set ge-1/0/6 unit 0 family bridge vlan-id-list 10-12
```

2. Configure the bridge domain parameters.

```
[edit bridge-domains]
user@host# set bd10 vlan-id 10
user@host# set bd10 enable-mac-move-action
user@host# set bd10 bridge-options interface ge-1/0/5.0 action-priority 1
user@host# set bd10 bridge-options interface ge-1/0/6.0 action-priority 5
user@host# set bd11 vlan-id 11
user@host# set bd11 enable-mac-move-action
user@host# set bd12 vlan-id 12
```

Results

From configuration mode, confirm your configuration by entering `show interfaces` and `show bridge-domains` commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@host# show interfaces
ge-1/0/4 {
    vlan-tagging;
    encapsulation flexible-ethernet-services;
    unit 10 {
        encapsulation vlan-bridge;
        vlan-id 10;
    }
    unit 11 {
        encapsulation vlan-bridge;
        vlan-id 11;
    }
}
ge-1/0/5 {
    unit 0 {
        family bridge {
            interface-mode trunk;
            vlan-id-list 10-12;
        }
    }
}
ge-1/0/6 {
    unit 0 {
        family bridge {
            interface-mode trunk;
            vlan-id-list 10-12;
        }
    }
}
```

```
user@host# show bridge-domains
bridge-domains {
    bd10 {
        vlan-id 10;
        bridge-options {
```

```
            interface ge-1/0/5.0 {
                action-priority 1;
            }
            interface ge-1/0/6.0 {
                action-priority 5
            }
        }
        enable-mac-move-action;
    }
    bd11 {
        vlan-id 11;
        enable-mac-move-action;
    }
    bd12 {
        vlan-id 12;
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

- Verifying That the Logical Interfaces Blocked Due to MAC Move Are Displayed | **48**

Verifying That the Logical Interfaces Blocked Due to MAC Move Are Displayed

Purpose

Ensure that the current set of logical interfaces blocked due to a MAC move, if any, are displayed.

Action

From operational mode, enter the `show l2-learning mac-move-buffer active` command.

```
user@host# show l2-learning mac-move-buffer active
MAC Address: 00:00:00:00:01:01, VLAN Id: 0
    Time Rec : 2012-06-25 06:23:41   Bridge Domain: bd10
```

```
    Prev IFL : ge-1/0/5.0           New IFL: ge-1/0/6.0
    IFBD     : ge-1/0/6.0:10        Blocked  : YES
```

Meaning

As a result of MAC move detection, one of the involved interface bridge domains will be blocked. The output shows that the ge-1/0/6 logical interface is blocked.

SEE ALSO

> *bridge-domains*
>
> Understanding Layer 2 Learning and Forwarding

Preventing Communication Among Customer Edge Devices as ACX Routers

In a bridge domain, when a frame is received from a CE interface, it is flooded to the other CE interfaces and all of the provider edge (PE) interfaces if the destination MAC address is not learned or if the frame is either broadcast or multicast. If the destination MAC address is learned on another CE device, such a frame is unicasted to the CE interface on which the MAC address is learned. This might not be desirable if the service provider does not want CE devices to communicate with each other directly.

To prevent CE devices from communicating directly, include the `no-local-switching` statement at the `[edit bridge-domains` *`bridge-domain-name`*`]` hierarchy level. Configure the logical interfaces in the bridge domain as core-facing (PE interfaces) by including the `core-facing` statement at the `[edit interfaces` *`interface-name`*`unit` *`logical-unit-number`* `family` *`family`*`]` hierarchy level to specify that the VLAN is physically connected to a core-facing ISP router and ensures that the network does not improperly treat the interface as a client interface. When specified, traffic from one CE interface is not forwarded to another CE interface.

For the `no-local-switching` option , integrated routing and bridging (IRB) configured on a bridge domain with this option enabled is not treated as a designated CE or PE interface. Traffic arriving from a CE or PE interface can navigate towards IRB and traffic that reaches in the input direction to the IRB can pass out of a CE or PE interface. The disabling of local switching achieves the functionality of split-horizon in a bridge domain. If no-local-switching is configured in a bridge domain, , then traffic cannot flow between CE and CE interfaces. This stoppage of trafic flow includes known unicast and multicast, unknown unicast and multicast, and broadcast traffic. However, traffic continues to be transmitted between CE and PE interfaces, and PE and PE interfaces..

Configuring MAC Learning Priority

You can configure MAC learning priority on interfaces so that MAC addresses are always learnt on the high priority interface.

If two interfaces receive the traffic with the same source MAC address, the MAC address is learnt on the high priority interface and the interface continues to forward the traffic. However, when an low priority interface receives the traffic from the same source MAC address, the traffic is discarded and will not be forwarded in the VLAN. MAC address move will not happen through the lower priority interface.

MAC address move is allowed when you configure the interfaces with the same MAC learning priority. When interfaces are not configured with MAC learning priority, then the default priority for each interface is 4.

In scenarios where you want the source MAC address to be learnt on a particular interface but still forward traffic received on other interfaces of the VLAN (without MAC move to the new interface), then you can configure persistent MAC learning on other interfaces. See Understanding and Using Persistent MAC Learning .

To configure MAC learning priority, use the `mac-learning-priority` configuration statement at the [`edit switch-options interface` *`interface-name`*] hierarchy level.

3

CHAPTER

Layer 2 Learning and Forwarding for Bridge Domains Functioning as Switches

Configure Layer 2 Learning and Forwarding for Bridge Domains Functioning as Switches

SUMMARY

IN THIS SECTION

Understanding Layer 2 Learning and Forwarding for Bridge Domains Functioning as Switches with Layer 2 Trunk Ports

Layer 2 learning is enabled by default. A set of bridge domains, configured to function as a switch with a Layer 2 trunk port, learns unicast media access control (MAC) addresses to avoid flooding packets to the trunk port.

NOTE: Traffic is not flooded back onto the interface on which it was received. However, because this "split horizon" occurs at a late stage, the packet statistics displayed by commands such as `show interfaces queue` will include flood traffic.

You can optionally disable Layer 2 learning for the entire set of bridge domains as well as modify the following Layer 2 learning and forwarding properties:

- Limit the number of MAC addresses learned from the Layer 2 trunk port associated with the set of bridge domains
- Modify the size of the MAC address table for the set of bridge domains
- Enable MAC accounting for the set of bridge domains

Configuring Bridge Domains as Switches for Layer 2 Trunk Ports

You can configure a set of bridge domains that are associated with a Layer 2 trunk port. The set of bridge domains function as a switch. Packets received on a trunk interface are forwarded within a bridge domain that has the same VLAN identifier. A trunk interface also provides support for IRB, which provides support for Layer 2 bridging and Layer 3 IP routing on the same interface.

To configure a Layer 2 trunk port and set of bridge domains, include the following statements:

```
[edit interfaces]
interface-name {
    unit number {
        family bridge {
            interface-mode access;
            vlan-id number;
        }
    }
}
interface-name {
    native-vlan-id number;
    unit number {
        family bridge {
            interface-mode trunk;
            vlan-id-list [ vlan-id-numbers ];
        }
    }
}
[edit bridge-domains]
bridge-domain-name {
    vlan-id number;
    vlan-id-list [ vlan-id-numbers ];
    . . . .
}
```

For **interface-mode trunk**, you can include the `vlan-id-list` statement.

You must configure a bridge domain and VLAN identifier for each VLAN associated with the trunk interface. You can configure one or more trunk or access interfaces at the `[edit interfaces]` hierarchy level. An access interface enables you to accept packets with no VLAN identifier. For more information about configuring trunk and access interfaces, see the Interfaces User Guide for Security Devices.

Limiting MAC Addresses Learned from a Layer 2 Trunk Port

You can configure a limit on the number of MAC addresses learned from a trunk port or from a specific trunk or access interface.

To limit the number of MAC addresses learned through a trunk port associated with a set of bridge domains, include the `interface-mac-limit` *`limit`* statement at the `[edit switch-options]` hierarchy level:

```
[edit]
switch-options {
    interface-mac-limit limit;
}
```

To limit the number of MAC addresses learned from a specific logical interface configured as an access interface or a trunk interface, include the `interface-mac-limit` *`limit`* statement at the `[edit switch-options interface` *`interface-name`*`]` hierarchy level:

```
[edit]
switch-options {
    interface interface-name {
        interface-mac-limit limit;
    }
}
```

The default value for the number MAC addresses that can be learned from a logical interface is 1024. You can specify a limit either for a set of bridge domains or for a specific logical interface in the range from 1 through 131,071. The value you configure for a specific logical interface overrides any value you specify for the set of bridge domains.

After the specified MAC address limit is reached, the default is for any incoming packets with a new source MAC address to be forwarded. You can specify that the packets be dropped for the entire virtual

switch after the MAC address limit is reached by including the `packet-action drop` statement at the `[edit switch-options interface-mac-limit limit]` hierarchy level:

```
[edit switch-options interface interface-name interface-mac-limit limit]
packet-action drop;
```

To specify that the packets be dropped from a specific logical interface in a set of bridge domains with a trunk port after the MAC address limit is reached, include the `packet-action drop` statement at the `[edit routing-instances routing-instance-name interface interface-name interface-mac-limit limit]` hierarchy level:

```
[edit routing-instances routing-instance-name interface interface-name interface-mac-limit limit]
packet-action drop;
```

Configuring the Size of the MAC Address Table for a Set of Bridge Domains

You can modify the size of the MAC address table for a set of bridge domains. The minimum you can configure is 16 addresses, and the maximum is 1,048,575 addresses. The default table size is 5120 addresses.

If the MAC table limit is reached, new addresses can no longer be added to the table. Unused MAC addresses are removed from the MAC address table automatically. This frees space in the table, allowing new entries to be added to the table.

To modify the size of the MAC table for a set of bridge domains, include the `mac-table-size` statement at the `[edit switch-options]` hierarchy level:

```
[edit switch-options]
mac-table-size limit;
```

Enabling MAC Accounting for a Set of Bridge Domains

By default, MAC accounting is disabled. You can enable packet counting for a set of bridge domains. After you enable packet accounting, the Junos OS maintains packet counters for each MAC address learned on the trunk port associated with the set of bridge domains.

To enable MAC accounting for a set of bridge domains, include the `mac-statistics` statement at the `[edit switch-options]` hierarchy level:

```
[edit switch-options]
mac-statistics;
```

Disabling MAC Learning for a Set of Bridge Domains

By default, MAC learning is enabled for a set of bridge domains. You can disable MAC learning for a set of bridge domains. Disabling dynamic MAC learning prevents the Layer 2 trunk port associated with the set of bridge domains from learning source and destination MAC addresses. When you disable MAC learning, source MAC addresses are not dynamically learned, and any packets sent to these source addresses are flooded into the switch.

To disable MAC learning for a set of bridge domains, include the `no-mac-learning` statement at the `[edit switch-options]` hierarchy level:

```
[edit switch-options]
no-mac-learning;
```

4

CHAPTER

Layer 2 Virtual Switches

Configuring Q-in-Q Tunneling on ACX Series

SUMMARY

IN THIS SECTION

Q-in-Q Tunneling on ACX Series Overview

Q-in-Q tunneling allows service providers to create a Layer 2 Ethernet connection between two customer sites. Providers can segregate different customers' VLAN traffic on a link (for example, if the customers use overlapping VLAN IDs) or bundle different customer VLANs into a single service VLAN. Service providers can use Q-in-Q tunneling to isolate customer traffic within a single site or to enable customer traffic flows across geographic locations.

Q-in-Q tunneling adds a service VLAN tag before the customer's 802.1Q VLAN tags. The Juniper Networks Junos operating system implementation of Q-in-Q tunneling supports the IEEE 802.1ad standard.

In Q-in-Q tunneling, as a packet travels from a customer VLAN (C-VLAN) to a service provider's VLAN (S-VLAN), another 802.1Q tag for the appropriate S-VLAN is added before the C-VLAN tag. The C-VLAN tag remains and is transmitted through the network. As the packet exits from the S-VLAN space, in the downstream direction, the S-VLAN 802.1Q tag is removed.

In ACX Series routers, you can configure Q-in-Q tunneling by explicitly configuring an input VLAN map with `push` function on customer facing interfaces in a bridge domain.

You can configure Q-in-Q tunneling on aggregated Ethernet interface by configuring input and output VLAN map.

Configuring Q-in-Q Tunneling on ACX Series

To configure Q-in-Q tunneling, you need to configure the logical interface connected to the customer network (user-to-network interfaces (UNI)) and the logical interface connected to the service provider network (network-to-network interface (NNI)).

The following is an example to configure a logical interface connected to a customer network:

```
[edit]
    interface ge-1/0/1 {
        flexible-vlan-tagging;
            encapsulation flexible-ethernet-services;
        unit 0 {
            encapsulation vlan-bridge;
                vlan-id-list 10-20;
                input-vlan-map {
                push;
                vlan-id 500;
                }
                output-vlan-map pop;
            }
    }
```

The following is an example to configure a logical interface connected to a service provider network:

```
[edit]
interface ge-1/0/2; {
    flexible-vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit 0 {
        encapsulation vlan-bridge;
        vlan-id 500;
    }
}
```

The following is an example to configure the bridge domain:

```
[edit]
bridge-domains {
    qnq-stag-500{
```

```
        interface ge-1/0/1;
        interface ge-1/0/2;
    }
}
```

You can configure Q-in-Q tunneling on aggregated Ethernet interface connected to the customer network (UNI) and the logical interface connected to the service provider network (NNI).

Configure Layer 2 Virtual Switches

SUMMARY

IN THIS SECTION

Understanding Layer 2 Virtual Switches

On MX Series routers only, you can group one or more bridge domains to form a virtual switch to isolate a LAN segment with its spanning-tree protocol instance and separate its VLAN ID space. A bridge domain consists of a set of logical ports that share the same flooding or broadcast characteristics. Like a virtual LAN, a bridge domain spans one or more ports of multiple devices. You can configure multiple

virtual switches, each of which operates independently of the other virtual switches on the routing platform. Thus, each virtual switch can participate in a different Layer 2 network.

You can configure a virtual switch to participate only in Layer 2 bridging and optionally to perform Layer 3 routing. In addition, you can configure one of three Layer 2 control protocols—Spanning-Tree Protocol, Rapid Spanning-Tree Protocol (RSTP), or Multiple Spanning-Tree Protocol (MSTP)—to prevent forwarding loops. For more information about how to configure Layer 2 logical ports on an interface, see the Junos OS Network Interfaces Library for Routing Devices.

In Junos OS Release 9.2 and later, you can associate one or more logical interfaces configured as trunk interfaces with a virtual switch. A trunk interface, or Layer 2 trunk port, enables you to configure a *logical interface* to represent multiple VLANs on the physical interface. Packets received on a trunk interface are forwarded within a bridge domain that has same VLAN identifier. For more information about how to configure trunk interfaces, see the Junos OS Network Interfaces Library for Routing Devices.

You can also configure Layer 2 forwarding and learning properties for the virtual switch as well as any bridge domains that belong to a virtual switch. .

For more information about configuring a routing instance for Layer 2 VPN, see the Junos OS VPNs Library for Routing Devices. .

Configuring a Layer 2 Virtual Switch

A Layer 2 virtual switch, which isolates a LAN segment with its spanning-tree protocol instance and separates its VLAN ID space, filters and forwards traffic only at the data link layer. Layer 3 routing is not performed. Each bridge domain consists of a set of logical ports that participate in Layer 2 learning and forwarding. A virtual switch represents a Layer 2 network.

Two main types of interfaces are used in virtual switch hierarchies:

- Layer 2 logical interface—This type of interface uses the VLAN-ID as a virtual circuit identifier and the scope of the VLAN-ID is local to the interface port. This type of interface is often used in service-provider-centric applications.
- Access or trunk interface—This type of interface uses a VLAN-ID with global significance. The access or trunk interface is implicitly associated with bridge domains based on VLAN membership. Access or trunk interfaces are typically used in enterprise-centric applications.

NOTE: The difference between access interfaces and trunk interfaces is that access interfaces can be part of one VLAN only and the interface is normally attached to an end-user device (packets are implicitly associated with the configured VLAN). In

contrast, trunk interfaces multiplex traffic from multiple VLANs and usually interconnect switches.

To configure a Layer 2 virtual switch, include the following statements:

```
[edit]
routing-instances {
    routing-instance-name (
        instance-type virtual-switch;
        bridge-domains {
            bridge-domain-name {
                domain-type bridge;
                interface interface-name;
                vlan-id (all | none | number); # Cannot be used with 'vlan-tags' statement
                vlan-id-list [ vlan-id-numbers ];
                vlan-tags outer number inner number; # Cannot be used with 'vlan-id' statement
            }
        }
        protocols {
            mstp {
                ...mstp-configuration ...
            }
        }
    }
}
```

To enable a virtual switch, you must specify **virtual-switch** as the **instance-type**.

For each bridge domain that you configure for the virtual switch, specify a ***bridge-domain-name***. You must also specify the value **bridge** for the `domain-type` statement.

For the `vlan-id` statement, you can specify either a valid VLAN identifier or the **none** or **all** options.

The **all** option is not supported with IRB.

NOTE: You do not have to specify a VLAN identifier for a bridge domain. However, you cannot specify the same VLAN identifier for more than one bridge domain within a virtual switch. Each bridge domain within a virtual switch must have a unique VLAN identifier.

NOTE: For a single bridge domain, you can include either the `vlan-id` statement or the `vlan-tags` statement, but not both. The `vlan-id` statement, `vlan-id-list` statement, and `vlan-tags` statement are mutually exclusive.

The `vlan-id-list` statement allows you to automatically create multiple bridge-domains for each vlan-id in the list.

To specify one or more logical interfaces to include in the bridge domain, specify an ***interface-name*** for an Ethernet interface you configured at the `[edit interfaces]` hierarchy level. For more information, see the Junos OS Network Interfaces Library for Routing Devices.

Configuring a Virtual Switch Routing Instance on MX Series Routers

On MX Series routers only, use the `virtual-switch` routing instance type to isolate a LAN segment with its spanning-tree instance and to separate its VLAN ID space. A bridge domain consists of a set of ports that share the same flooding or broadcast characteristics. Each virtual switch represents a Layer 2 network. You can optionally configure a virtual switch to support Integrated Routing and Bridging (IRB), which facilitates simultaneous Layer 2 bridging and Layer 3 IP routing on the same interface. You can also configure Layer 2 control protocols to provide loop resolution. Protocols supported include the Spanning-Tree Protocol (STP), Rapid Spanning-Tree Protocols (RSTP), Multiple Spanning-Tree Protocol (MSTP), and VLAN Spanning-Tree Protocol (VSTP).

To create a routing instance for a virtual switch, include at least the following statements in the configuration:

```
[edit]
routing-instances {
    routing-instance-name
        instance-type virtual-switch;
        bridge-domains {
            bridge-domain-name {
                domain-type bridge;
                interface interface-name;
                vlan-id (all | none | number);
                vlan-tags outer number inner number;
            }
        }
        protocols {
            (rstp | mstp | vstp) {
```

```
                ...stp-configuration ...
            }
        }
    }
}
```

For more information about configuring virtual switches, see Configuring a Layer 2 Virtual Switch .

Configuring VPLS Ports in a Virtual Switch

In Junos OS Release 9.3 and later, you can configure VPLS ports in a virtual switch so that the logical interfaces of the Layer 2 bridge domains in the virtual switch can handle VPLS routing instance traffic. VPLS configuration no longer requires a dedicated routing instance of type **vpls**. Packets received on a Layer 2 trunk interface are forwarded within a bridge domain that has the same VLAN identifier.

A trunk interface is implicitly associated with bridge domains based on VLAN membership. Whereas access interfaces can be part of one VLAN only, trunk interfaces multiplex traffic from multiple VLANs and usually interconnect switches. A Layer 2 trunk port also supports IRB.

To configure VPLS ports in a virtual switch, perform the following tasks:

1. To configure the Layer 2 trunk ports that you will associate with the bridge domains in the virtual switch, include the following statements in the configuration:

```
[edit]
interfaces {
    interface-name {
        unit logical-unit-number { # Call this 'L2-trunk-port-A'
            family bridge {
                interface-mode trunk;
                vlan-id-list [ vlan-id-numbers ] ; # Trunk mode VLAN membership for this
interface
            }
        }
    }
    .
    .
    .
    interface-name {
        unit logical-unit-number { # Call this 'L2-trunk-port-B'
            family bridge {
```

```
                interface-mode trunk;
                vlan-id-list [ vlan-id-numbers ] ; # Trunk mode VLAN membership for this
interface
            }
        }
    }
}
```

To configure a logical interface as a trunk port, include the `interface-mode` statement and the **trunk** option at the `[edit interfaces` *`interface-name`* `unit` *`logical-unit-number`* `family bridge]` hierarchy level.

To configure all the VLAN identifiers to associate with a Layer 2 trunk port, include the `vlan-id-list [` *`vlan-id-numbers`* `]` statement at the `[edit interfaces` *`interface-name`* `unit` *`logical-unit-number`* `family bridge]` hierarchy level.

Each of the logical interfaces "***L2-trunk-port-A***" and "***L2-trunk-port-B***" accepts packets tagged with any VLAN ID specified in the respective `vlan-id-list` statements.

2. To configure a virtual switch consisting of a set of bridge domains that are associated with one or more logical interfaces configured as a trunk ports, include the following statements in the configuration:

```
[edit]
routing-instance {
    routing-instance-name
        instance-type virtual-switch;
        interface L2-trunk-port-A; # Include one trunk port
        interface L2-trunk-port-B; # Include the other trunk port
        bridge-domains {
            bridge-domain-name-0 {
                domain-type bridge;
                vlan-id number;
            }
            bridge-domain-name-1 {
                domain-type bridge;
                vlan-id number;
            }
        }
        protocols {
            vpls {
                vpls-id number;
                ... vpls-configuration ...
            }
```

```
        }
    }
}
```

To begin configuring a virtual switch, include the `instance-type` statement and the **virtual-switch** option at the `[edit routing-instances routing-instance-name]` hierarchy level.

To configure a virtual switch consisting of a set of bridge domains that are associated with one or more logical interfaces configured as a trunk ports, you must identify each logical interface by including the `interface interface-name` statement at the `[edit routing-instances routing-instance-name]` hierarchy level.

For each VLAN configured for a trunk port, you must configure a bridge-domain that includes the trunk port logical interface and uses a VLAN identifier within the range carried by that trunk interface. To configure, include the **domain-type bridge**, **vlan-id *number***, and statements at the `[edit routing-instances routing-instance-name bridge-domain bridge-domain-name]` hierarchy level.

Configuring a Layer 2 Virtual Switch with a Layer 2 Trunk Port

You can associate one or more Layer 2 trunk interfaces with a virtual switch. A Layer 2 trunk interface enables you to configure a logical interface to represent multiple VLANs on the physical interface. Within the virtual switch, you configure a bridge domain and VLAN identifier for each VLAN identifier configured on the trunk interfaces. Packets received on a trunk interface are forwarded within a bridge domain that has the same VLAN identifier. Each virtual switch you configure operates independently and can participate in a different Layer 2 network.

A virtual switch configured with a Layer 2 trunk port also supports IRB within a bridge domain. IRB provides simultaneous support for Layer 2 bridging and Layer 3 IP routing on the same interface. Only an interface configured with the `interface-mode (access | trunk)` statement can be associated with a virtual switch. An access interface enables you to accept packets with no VLAN identifier. For more information about configuring trunk and access interfaces, see the Junos OS Network Interfaces Library for Routing Devices.

In addition, you can configure Layer 2 learning and forwarding properties for the virtual switch.

To configure a virtual switch with a Layer 2 trunk interface, include the following statements:

```
[edit]
routing-instances {
    routing-instance-name {
        instance-type virtual-switch;
        interface interface-name;
```

```
        bridge-domains {
            bridge-domain-name {
                vlan-id number;
            }
        }
    }
}
```

NOTE: You must configure a bridge domain and VLAN identifier for each VLAN identifier configured for the trunk interface.

Layer 2 trunk ports are used in two distinct types of virtual switch configuration. One method is called service provider style and the other is called enterprise style. The two methods can be confusing because both methods involve configuring interfaces known as trunk interfaces. However, both types of configuration are distinct.

Service provider style and enterprise style each have benefits and drawbacks.

- Service provider style—Offers more control, but requires more care in configuration. Service providers can use all bridging features in any shape or size, but for large bridged designs, customization requirements quickly grow.
- Enterprise style—Offers a single Layer 2 network connected by simple bridges. Easier to use, but more limited in function. Configuration is simple and straightforward and condensed.

NOTE: The terms “service provider style” and “enterprise style” do not imply any limitations based on organization type or size. Any large enterprise may use service-provider-style configurations and a small regional service provider is free to use enterprise style. The differences apply only to the configuration styles.

The easiest way to understand the differences in configuration of the two styles is to compare them using the same interfaces and VLAN IDs.

You can configure multiple bridge domains between the same pair of Ethernet interfaces, for example, `xe-0/0/1` and `xe-0/0/2`. If there are two bridge domains needed, you can configure one bridge domain as `VLAN-100` and the other as `VLAN-200`. However, the configuration requirements are different when implementing service provider style or enterprise style. Here is a look at both styles using the same interfaces and VLANs.

Service provider style involves configuring the values for three main parameters, plus the bridge domains to connect them:

- VLAN tagging—Configure the bridged physical interfaces with `vlan-tagging` to allow them to operate in IEEE 802.1Q mode, also known as a trunk interface.
- Extended VLAN Bridge—Configure the physical interface with the encapsulation statement type `extended-vlan-bridge` to allow bridging on each logical interface.
- Logical unit—Configure a logical unit for each bridged VLAN ID. In most cases, you configure the unit number to be the same as the VLAN ID (that is, unit 100 = VLAN ID 100).
- Bridge domains—Configure the VLAN bridge domains to associate the logical interfaces with the correct VLAN IDs.

Here is the service provider style configuration showing two interfaces used for bridging across two bridge domains, VLAN ID 100 and 200.

```
[edit]
interfaces {
    xe-0/0/1 {
        vlan-tagging;
        encapsulation extended-vlan-bridge;
        unit 100 {
            vlan-id 100;
        }
        unit 200 {
            vlan-id 200;
        }
    }
    xe-0/0/2 {
        vlan-tagging;
        encapsulation extended-vlan-bridge;
        unit 100 {
            vlan-id 100;
        }
        unit 200 {
            vlan-id 200;
        }
    }
}

bridge-domains {
    VLAN-100 {
        vlan-id 100;
        interface xe-0/0/1.100;
```

```
        interface xe-0/0/2.100;
    }
    VLAN-200 {
        vlan-id 200;
        interface xe-0/0/1.200;
        interface xe-0/0/2.200;
    }
}
```

Note that each physical interface has VLAN tagging enabled as well as extended VLAN bridge encapsulation. There are many more parameters that can be configured in service provider style.

In contrast, enterprise style involves configuring the values for three *different* parameters, plus the bridge domains to connect them:

- Family— Configure each bridged physical interface with the family type `bridge`.
- Interface mode—Configure logical interface so that the physical interface operates as either an untagged access port (not shown in this topic) or as an IEEE 801Q `trunk`.
- VLAN ID—Configure each logical interface with a VLAN ID to determine with which bridge the interface belongs.
- Bridge domain—Configure the VLAN bridge domains to associate with the correct VLAN IDs.

NOTE: Enterprise style is simpler than the service provider style. Enterprise style automatically places interfaces in bridge domains when the configuration is committed.

Here is the enterprise style configuration showing the same two interfaces used for bridging across the same two bridge domains, VLAN ID 100 and 200.

```
[edit]
interfaces {
    xe-0/0/1 {
        unit 0 {
            family bridge {
                interface-mode trunk;
                vlan-id-list [ 100 200 ];
            }
        }
    }
    xe-0/0/2 {
        unit 0 {
```

```
            family bridge {
                interface-mode trunk;
                vlan-id-list [ 100 200 ];
            }
        }
    }
}

bridge-domains {
    VLAN-100 {
        vlan-id 100;
    }
    VLAN-200 {
        vlan-id 200;
    }
}
```

In exchange for simplicity, enterprise style does not allow you to configure VLAN tagging options or encapsulation type. You do not create a separate logical interface for each VLAN ID.

NOTE: You can configure more parameters in each style. These further parameters are beyond the scope of this basic configuration topic.

Configuring Integrated Routing and Bridging for a Bridge Domain in a Layer 2 Virtual Switch

Integrated routing and bridging (IRB) provides simultaneous support for Layer 2 bridging and Layer 3 IP routing on the same interface. IRB enables you to route local packets to another routed interface or to another bridge domain that has a Layer 3 protocol configured. You configure a logical routing interface by including the `irb` statement at `[edit interfaces]` hierarchy level and include that interface in the bridge domain. For more information about how to configure a routing interface, see the Junos OS Network Interfaces Library for Routing Devices.

NOTE: You can include only one routing interface in a bridge domain.

To configure a virtual switch with IRB support, include the following statements:

```
[edit]
routing-instances {
    routing-instance-name {
        instance-type virtual-switch;
        bridge-domains {
            bridge-domain-name {
                domain-type bridge;
                interface interface-name;
                routing-interface routing-interface-name;
                vlan-id (none | number);
                vlan-tags outer number inner number;
            }
        }
    }
}
```

To enable a virtual switch, you must specify **virtual-switch** as the **instance-type**. The `instance-type virtual-switch` statement is not supported at the `[edit logical-systems logical-system-name]` hierarchy level.

For each bridge domain that you configure for the virtual switch, specify a ***bridge-domain-name***. You must also specify the value **bridge** for the `domain-type` statement.

For the `vlan-id` statement, you can specify either a valid VLAN identifier or the **none** option.

NOTE: For a single bridge domain, you can include either the `vlan-id` statement or the `vlan-tags` statement, but not both.

To include one or more logical interfaces in the bridge domain, specify the ***interface-name*** for each Ethernet interface to include that you configured at the `[edit interfaces irb]` hierarchy level.

To associate a routing interface with a bridge domain, include the `routing-interface routing-interface-name` statement and specify a ***routing-interface-name*** you configured at the `[edit interfaces irb]` hierarchy level. You can configure only one routing interface for each bridge domain. For more information about how to configure logical and routing interfaces, see the Junos OS Network Interfaces Library for Routing Devices.

NOTE: If you configure a routing interface to support IRB in a bridge domain, you cannot use the **all** option for the `vlan-id` statement.

Configuring Integrated Routing and Bridging in ACX Series

Integrated routing and bridging (IRB) provides simultaneous support for Layer 2 bridging and Layer 3 routing on the same interface. IRB enables you to route packets to another routed interface or to another bridge domain that has an IRB interface configured. You configure a logical routing interface by including the `irb` statement at the `[edit interfaces]` hierarchy level and include that interface in the bridge domain. For more information about how to configure a routing interface, see the Junos OS Network Interfaces Library for Routing Devices.

NOTE: You can include only one routing interface in a bridge domain.

The following are the list of features supported for IRB:

- Family `inet`, `inet6`, and `iso` are supported on an IRB interface.
- Routing protocols supported on an IRB interface are BGP, ISIS, OSPF, RIP, IGMP, and PIM.
- DHCP Relay with option 82 is supported on an IRB interface.
- IRB can be added in a VRF routing instance.
- VRRP is supported on an IRB inteface.
- Bidirectional Forwarding Detection (BFD) protocol is supported on an IRB interface.
- The following Class-of-Service configurations are supported on an IRB interface:
  - Fixed classifier can be applied on an IRB logical interface.
  - Firewall filters (multifield filter) can be used to assign forwarding class and loss priority. You should define a family inet or inet6 filter and apply it as the input filter on an IRB logical interface under family inet.

    NOTE: `physical-interface-filter` is not supported for family inet6 filter on IRB logical interface.

  - dscp, inet-precedence, ieee-802.1, and ieee-802.1ad values can be rewritten.

ACX routers do not support MPLS families on IRB.

IRB can be configured under the following hierarchies:

- `[edit intefaces irb interface_type]` hierarchy level
  - disable—Disables the interface

  - gratuitous-arp-reply—Enables gratuitous ARP reply
  - hold-time—Hold time for link up and link down
  - mtu—Maximum transmit packet size (256..9192)
  - no-gratuitous-arp-reply—Does not enable gratuitous ARP reply
  - no-gratuitous-arp-request—Ignores gratuitous ARP request
- [`edit interfaces irb.`*`unit`* `family (inet | inet6 | iso)`] hierarchy level
- [`edit bridge-domains routing-interface interface irb.`*`unit`*] hierarchy level
- [`edit routing-instances instance-type vrf`] hierarchy level
- [`edit protocols (`*`bgp | isis | ospf | rip | igmp | pim`*`) interface irb.`*`unit`*] hierarchy level
- [`edit class-of-service interfaces irb`]] hierarchy level

In ACX5048 and ACX5096 routers, you can configure IRB at the [`edit vlans `*`vlan-name`*`] l3-interface irb.`*`unit`*; level.

NOTE: The Layer 2 CLI configurations and show commands for ACX5048 and ACX5096 routers differ compared to other ACX Series routers. For more information, see *Layer 2 Next Generation Mode for ACX Series*.

To configure a bridge domain with IRB support, include the following statements:

```
[edit]
bridge-domains {
    bridge-domain-name {
        domain-type bridge;
        interface interface-name;
        routing-interface routing-interface-name;
        vlan-id (none | number);
        vlan-tags outer number inner number;
    }
}
```

For each bridge domain that you configure, specify a ***bridge-domain-name***. You must also specify the value **bridge** for the `domain-type` statement.

For the `vlan-id` statement, you can specify either a valid VLAN identifier or the **none** option.

The `vlan-tags` statement enables you to specify a pair of VLAN identifiers; an **outer** tag and an **inner** tag.

NOTE: For a single bridge domain, you can include either the `vlan-id` statement or the `vlan-tags` statement, but not both.

To include one or more logical interfaces in the bridge domain, specify the ***interface-name*** for each Ethernet interface to include that you configured at the `[edit interfaces]` hierarchy level.

NOTE: A maximum of 4000 active logical interfaces are supported on a bridge domain configured for Layer 2 bridging.

To associate a routing interface with a bridge domain, include the `routing-interface` *`routing-interface-name`* statement and specify a ***routing-interface-name*** you configured at the `[edit interfaces irb]` hierarchy level. You can configure only one routing interface for each bridge domain. For more information about how to configure logical and routing interfaces, see the Junos OS Network Interfaces Library for Routing Devices.

In Junos OS Release 9.0 and later, IRB interfaces are supported for multicast snooping. For more information about multicast snooping, see the Junos OS Multicast Protocols User Guide.

NOTE: When you configure multiple IRB logical interfaces, all the IRB logical interfaces share the same MAC address.

The following is a sample configuration for IRB over bridge domain:

```
[edit]
interfaces {
    ge-1/0/0 {
        encapsulation flexible-ethernet-services;
            flexible-vlan-tagging;
            unit 0 {
                encapsulation vlan-bridge;
                vlan-id 100;
            }
        }
    }
    ge-1/0/1 {
        encapsulation flexible-ethernet-services;
            flexible-vlan-tagging;
            unit 0 {
```

```
                encapsulation vlan-bridge;
                vlan-id 100;
            }
        }
    }
    irb {
        unit 0 {
            family inet {
                address 10.0.1.2/24 {
            }
        }
    }
}
bridge-domains {
    bd {
        domain-type bridge;
        vlan-id none;
        interface ge-1/0/0.0;
        interface ge-1/0/1.0;
        routing-interface irb.0;
    }
}
```

1

PART

Configuration Statements and Operational Commands

Junos CLI Reference Overview

We've consolidated all Junos CLI commands and configuration statements in one place. Learn about the syntax and options that make up the statements and commands and understand the contexts in which you'll use these CLI elements in your network configurations and operations.

- Junos CLI Reference

Click the links to access Junos OS and Junos OS Evolved configuration statement and command summary topics.

- Configuration Statements
- Operational Commands