CONFIGURATION GUIDE

Extreme SLX-OS Network Packet Broker Configuration Guide, 18s.1.01

Supporting the ExtremeSwitching SLX 9140 and SLX 9240 Switches

9035255-01
November 2018

Legal Notice

Extreme Networks, Inc. reserves the right to make changes in specifications and other information contained in this document and its website without prior notice. The reader should in all cases consult representatives of Extreme Networks to determine whether any such changes have been made.

The hardware, firmware, software or any specifications described or referred to in this document are subject to change without notice.

Trademarks

Extreme Networks and the Extreme Networks logo are trademarks or registered trademarks of Extreme Networks, Inc. in the United States and/or other countries.

All other names (including any product names) mentioned in this document are the property of their respective owners and may be trademarks or registered trademarks of their respective companies/owners.

For additional information on Extreme Networks trademarks, please see: www.extremenetworks.com/company/legal/trademarks

Software Licensing

Some software files have been licensed under certain open source or third-party licenses. End-user license agreements and open source declarations can be found at: www.extremenetworks.com/support/policies/software-licensing

Support

For product support, phone the Global Technical Assistance Center (GTAC) at 1-800-998-2408 (toll-free in U.S. and Canada) or +1-408-579-2826. For the support phone number in other countries, visit: http://www.extremenetworks.com/support/contact/

For product documentation online, visit: https://www.extremenetworks.com/documentation/

Contents

Preface

This section discusses the conventions used in this guide, ways to provide feedback, additional help, and other Extreme Networks® publications.

Conventions

This section discusses the conventions used in this guide.

Notes, cautions, and warnings

Notes, cautions, and warning statements may be used in this document. They are listed in the order of increasing severity of potential hazards.

NOTE

A Note provides a tip, guidance, or advice, emphasizes important information, or provides a reference to related information.

ATTENTION

An Attention statement indicates a stronger note, for example, to alert you when traffic might be interrupted or the device might reboot.

CAUTION

A Caution statement alerts you to situations that can be potentially hazardous to you or cause damage to hardware, firmware, software, or data.

DANGER

A Danger statement indicates conditions or situations that can be potentially lethal or extremely hazardous to you. Safety labels are also attached directly to products to warn of these conditions or situations.

Text formatting conventions

Text formatting conventions such as boldface, italic, or Courier font may be used to highlight specific words or phrases.

Format	Description
bold text	Identifies command names.
	Identifies keywords and operands.
	Identifies the names of GUI elements.
	Identifies text to enter in the GUI.
italic text	Identifies emphasis.
	Identifies variables.
	Identifies document titles.

Format	Description
`Courier font`	Identifies CLI output. Identifies command syntax examples.

Command syntax conventions

Bold and italic text identify command syntax components. Delimiters and operators define groupings of parameters and their logical relationships.

Convention	Description
bold text	Identifies command names, keywords, and command options.
italic text	Identifies a variable.
[ ]	Syntax components displayed within square brackets are optional. Default responses to system prompts are enclosed in square brackets.
{ **x** | **y** | **z** }	A choice of required parameters is enclosed in curly brackets separated by vertical bars. You must select one of the options.
x | **y**	A vertical bar separates mutually exclusive elements.
< >	Nonprinting characters, for example, passwords, are enclosed in angle brackets.
...	Repeat the previous element, for example, *member*[*member*...].
\	Indicates a "soft" line break in command examples. If a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash.

Documentation and Training

To find Extreme Networks product guides, visit our documentation pages at:

Current Product Documentation	www.extremenetworks.com/documentation/
Archived Documentation (for earlier versions and legacy products)	www.extremenetworks.com/support/documentation-archives/
Release Notes	www.extremenetworks.com/support/release-notes
Hardware/Software Compatibility Matrices	https://www.extremenetworks.com/support/compatibility-matrices/
White papers, data sheets, case studies, and other product resources	https://www.extremenetworks.com/resources/

Open Source Declarations

Some software files have been licensed under certain open source licenses. More information is available at: www.extremenetworks.com/support/policies/open-source-declaration/.

Training

Extreme Networks offers product training courses, both online and in person, as well as specialized certifications. For more information, visit www.extremenetworks.com/education/.

Getting Help

If you require assistance, contact Extreme Networks using one of the following methods:

Extreme Portal	Search the GTAC (Global Technical Assistance Center) knowledge base, manage support cases and service contracts, download software, and obtain product licensing, training, and certifications.
The Hub	A forum for Extreme Networks customers to connect with one another, answer questions, and share ideas and feedback. This community is monitored by Extreme Networks employees, but is not intended to replace specific guidance from GTAC.
Call GTAC	For immediate support: 1-800-998-2408 (toll-free in U.S. and Canada) or +1 408-579-2826. For the support phone number in your country, visit: www.extremenetworks.com/support/contact

Before contacting Extreme Networks for technical support, have the following information ready:

- Your Extreme Networks service contract number and/or serial numbers for all involved Extreme Networks products
- A description of the failure
- A description of any action(s) already taken to resolve the problem
- A description of your network environment (such as layout, cable type, other relevant environmental information)
- Network load at the time of trouble (if known)
- The device history (for example, if you have returned the device before, or if this is a recurring problem)
- Any related RMA (Return Material Authorization) numbers

Subscribing to Service Notifications

You can subscribe to email notifications for product and software release announcements, Vulnerability Notices, and Service Notifications.

1. Go to www.extremenetworks.com/support/service-notification-form.
2. Complete the form with your information (all fields are required).
3. Select the products for which you would like to receive notifications.

   NOTE
   You can modify your product selections or unsubscribe at any time.

4. Click **Submit**.

Providing Feedback to Us

Quality is our first concern at Extreme Networks, and we have made every effort to ensure the accuracy and completeness of this document. We are always striving to improve our documentation and help you work better, so we want to hear from you! We welcome all feedback but especially want to know about:

- Content errors or confusing or conflicting information.
- Ideas for improvements to our documentation so you can find the information you need faster.
- Broken links or usability issues.

If you would like to provide feedback to the Extreme Networks Information Development team, you can do so in two ways:

- Use our short online feedback form at https://www.extremenetworks.com/documentation-feedback/.

- Email us at documentation@extremenetworks.com.

Please provide the publication title, part number, and as much detail as possible, including the topic heading and page number if applicable, as well as your suggestions for improvement.

About This Document

What's new in this document

The following table includes descriptions of changes to this guide for the current release.

TABLE 1 Changes in Extreme SLX-OS Network Packet Broker Configuration Guide, 18s.1.01

Feature	Description	Described in
Internal Loopback	A software feature to replicate traffic for service-port chaining and other implementations.	Internal Loopback on page 51
Onboard Packet Capture	The onboard packet capture feature captures frames that ingress or egress through a port in a data plane.	Onboard Packet Capture on page 55

Supported hardware and software

In those instances in which procedures or parts of procedures documented here apply to some devices but not to others, this guide identifies exactly which devices are supported and which are not.

Although many different software and hardware configurations are tested and supported by Extreme Networks, Inc. for this SLX-OS release, documenting all possible configurations and scenarios is beyond the scope of this document.

The following hardware platforms are supported by this release:

- ExtremeSwitching SLX 9140
- ExtremeSwitching SLX 9240

NOTE

Some of the commands in this document use a slot/port designation. Because the SLX 9140 and the SLX 9240 do not contain line cards, the slot designation must always be "0" (for example, 0/1 for port 1).

Basics of Network Packet Broker

NPB overview

A Network Packet Broker (NPB) provides a collection of monitoring tools with access to traffic across the network.

When you configure and reboot an SLX-OS device into NPB mode, the device can function as a Network Packet Broker.

The SLX-OS NPB implementation includes the following features:

- Aggregation: Traffic on multiple ingress interfaces is aggregated and forwarded on a single egress interface ("many to one") to one monitoring tool.
- Replication: Traffic on a single ingress interface is replicated and forwarded on multiple egress interfaces ("one to many") to multiple monitoring tools.
- Load balancing: Aggregation traffic and replication traffic are load-balanced across the members of an egress port-channel interface. Symmetric load balancing is the only type of load balancing supported.
- Header modification: Header stripping can reduce packet overhead, increasing the efficiency of the security and monitoring tools. The dropping of GPRS Tunneling Protocol (GTP) frames that encapsulate HTTPs packets is also supported.
- Timestamping: With Precision Time Protocol (PTP) accuracy, ingress and egress timestamps aid analysis of congestion and security issues. For details, refer to the "Precision Time Protocol (PTP)" section of the *Extreme SLX-OS Management Configuration Guide for SLX 9140 and SLX 9240*.

The following table indicates which interface types are supported for the aggregation, replication, load balancing, and timestamping features:

TABLE 2 Ingress and egress interface types

Feature	Ingress interfaces	Egress interfaces
Aggregation	Multiple physical, port-channel, or mixed interfaces	One physical or port-channel interface
Replication	One physical or port-channel interface	A TVF domain, including multiple physical, port-channel, or mixed interfaces
Load balancing	Not supported	One port-channel interface
Timestamping	Physical and port-channel interfaces	Physical and port-channel interfaces

Route maps under NPB

A route map is a container for one or more numbered permit or deny stanzas; each stanza contains a sequence of statements.

Evaluation of route maps consists of a list scan, from the lowest-numbered stanza to the highest-numbered stanza. Within each stanza, statements are evaluated in order. The following technologies are some that use route maps:

- BGP and OSPF protocols. For details, refer to the *Extreme SLX-OS Layer 3 Routing Configuration Guide for SLX 9140 and SLX 9240*.
- Policy-based routing (PBR) (not supported)
- Network Packet Broker (NPB)

Route-map syntax and evaluation vary with the technology. However, "match" statements are common to all route-map implementations. Upon a match, the actions specified in the match statement and in the remaining statements of that stanza are implemented. The list scan ends without examining higher-numbered stanzas.

Route-map "set" statements are also supported for the mentioned technologies. Upon a match, set statements perform an action on the matched traffic. The action varies with the technology. The following table compares route-map features and functionality for the various technologies.

TABLE 3 Route-map comparison

Feature	NPB	BGP	PBR (not supported)
Traffic routing or forwarding	Yes	Yes	Yes
Traffic redistribution	No	Yes	No
Route-attribute modification	No	Yes	No
Stanzas	One or more permit or deny stanzas	One or more permit or deny stanzas	One or more permit or deny stanzas
Match statements	One **match { mac | ip | ipv4 | uda } address acl** statement	One or more supported **match** statements	One or more **match { ip | ipv4 } address acl** statements
Set statements (supported only in permit stanzas)	**set interface** or **set next-hop-tvf-domain**	0, 1, or multiple **set** statements	0, 1, or multiple **set** statements
Continue statements	No	0 or 1 **continue** statements	No
Interfaces	Physical and port-channel interfaces	Device-level	Layer 3 interfaces

NPB configuration guidelines

Follow these guidelines when implementing Network Packet Broker (NPB):

- NPB requires a license. For details, refer to the *Extreme SLX-OS Software Licensing Guide for SLX 9140 and SLX 9240*.
- A system reboot is required when moving from default system mode to NPB mode and conversely.
- Most Layer 2 and Layer 3 configurations are not supported in NPB mode. Although implementing them does not generate error messages, do not configure them in NPB mode.

- Only one route map can be applied per interface as NPB policy. However, that route map can have multiple stanzas.
- You can apply a given route map on multiple interfaces.
- The only supported match statements are **match { mac | ip | ipv6 | uda } address acl** statements. Other match statements are ignored.
- The only supported set statements are **set interface** and **set next-hop-tvf-domain**. Other set statements are ignored.
- If a stanza does not contain a match statement, "match any" is implied.
- The only supported egress interfaces are physical interfaces, port-channel interfaces, and TVF domains.
- If a physical interface is part of a port-channel, a **set interface** statement on that interface is ignored.

Forwarding priority guidelines

The following guidelines determine forwarding priority in an NPB route-map:

- In general, forwarding priority is determined by the first match in the lowest-numbered stanza. For exceptions, refer to Priority among L2 and L3 match acl statements on page 15.
- The order in which the egress interfaces are configured is also the order for configuring forwarding.
- If a route map has multiple egress interfaces, the first egress interface ready for forwarding is used.
  - A physical interface is considered ready for forwarding if its link is up.
  - Port-channels and TVF domains are considered ready if at least one member port link is up.
- If the current egress interface loses its Ready status, then the next configured egress interface that is ready for forwarding is used.
- If a Down egress interface becomes Ready and if it has higher priority than the currently selected egress interface, the higher priority egress interface replaces it.
- If none of the configured egress interfaces are ready, the traffic is dropped.

Setting NPB as the system mode

Before enabling Network Packet Broker (NPB), you must set NPB as the system mode.

Accessing NPB mode requires a license. For details, refer to the *Extreme SLX-OS Software Licensing Guide for SLX 9140 and SLX 9240*.

1. Enter **configure terminal** to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter **hardware** to access hardware configuration mode.

   ```
   device(config)# hardware
   ```

3. Enter **system-mode npb** to change the system mode to NPB.

   ```
   device(config-hardware)# system-mode npb
   %Warning: To activate the new system-mode config, please reboot the system using 'reload system'.
   ```

   To return to default system mode from NPB system mode, enter **system-mode default** rather than **system-mode npb**.

4. Enter **end** to access privileged EXEC mode.

```
device(config-hardware)# end
```

5. To reboot the system, effecting the system mode change, enter **reload system**.

```
device# reload system
Warning: This operation will cause the chassis to reboot and requires all existing telnet,
secure telnet and SSH sessions to be restarted.
Unsaved configuration will be lost.
Please run `copy running-config startup-config` to save the current configuration if not done
already.
Are you sure you want to reboot the chassis [y/n]?
```

6. Press **y** and then **Enter**.

ACLs and UDAs under NPB

Stanza and ACL permit and deny keywords

Both route-map stanzas and access-control lists (ACLs) have **permit** and **deny** keywords.

> NOTE
> In this context, "ACL" includes standard, extended, and user-defined ACLs (UDAs).

In NPB mode, you use ACLs only within route-maps, to exclude certain traffic flows from set statements.

Both NPB and PBR route-maps contain **match { mac | ip | ipv4 } address acl** statements. (NPB route-maps can also contain **match uda address acl** statements.) However, permit and deny rules in ACLs applied to route maps function differently than rules in the security ACLs discussed in the *Extreme SLX-OS Security Configuration Guide* :

- In security ACLs, permit rules allow packets and deny rules drop packets.
- In ACLs applied to PBR route-maps, permit and deny rules specify criteria for route-map decisions.
- In ACLs applied to NPB route-maps, permit and deny rules are mechanisms to exclude certain traffic flows from set statements.

The following table describes the interactions between route-map permit and deny stanzas; and permit and deny rules in ACLs applied to those stanzas by **match { mac | ip | ipv6 | uda } address acl** statements.

TABLE 4 Route-map stanza and ACL permit and deny interactions

Stanza	ACL rule	Resulting TCAM action
Permit	Permit	The **set** statement or statements are applied.
Permit	Deny	Packets that match a **deny** keyword are denied from using the stanza **set** statement. The packet is routed as normal.
Deny	Permit	No action is taken; the packet is routed as normal.
Deny	Deny	No action is taken; the packet is routed as normal.

Priority among L2 and L3 match acl statements

In general, forwarding priority is determined by the first match in the lowest-numbered stanza. However, there are exceptions to this priority, if a route map contains match statements from multiple groups:

- Layer 2: **match mac address acl**

- Layer 3: **match { ip | ipv6 } address acl**
- User-defined ACLs: **match uda acl**

Priority among Layer 2, Layer 3, and UDA match statements is as follows:

- If the match within one ACL is on a permit rule and the match in another ACL is on a deny rule, the permit match is accepted and the deny match is ignored.
- If there are multiple Layer 3 matches, the first match in the lowest-numbered stanza is accepted and other matches are ignored.
- If multiple matches—in different ACL types—are on permit rules, only the match in the highest-preference type is implemented; lower-preference matches are ignored. The preference order is Layer 3 > Layer 2 > UDA.

Creating ACLs for NPB

For NPB traffic aggregation and traffic replication, an access-control list (ACL) or user-defined ACL (UDA)—included in a route-map—is required.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **{ mac | ip | ipv6 } access-list** command to create an ACL.

```
device(config)# ip access-list standard aclNPB_01
```

3. Create one or more permit or deny rules.

```
device(conf-ipacl-std)# permit host 192.1.1.1 count
```

NOTE
A deny rule specifies that a matching packet is denied from using the **set** statement.

Specifying an IPv6 lookup-profile

For flexibility with IPv6 header-elements specified in ACLs, you can configure which half of SIP and DIP addresses are copied into the token header-buffer.

Each forwarded frame has a token with a 100-byte header buffer for storing header data. Fields copied into this token header-buffer are available for lookup (to make forwarding decisions). Because of this small buffer size, copying the entire 128-bit SIP and DIP addresses is not supported. You can configure the IPv6 lookup profile, specifying which half of SIP and DIP addresses are copied into the buffer:

- (Default) Host ID (bits 64-127)
- Network ID (bits 0-63)

Configuration guidelines:

- The software does not prevent you from specifying a full 128-bit IPV6 address while configuring an IPv6 ACL. However, only the half configured in the IPv6 lookup profile is matched.
- Before you change lookup-profile, you need to evaluate the impact of the change on current IPv6 ACLs.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter **hardware** to access hardware configuration mode.

```
device(config)# hardware
```

3. Enter the **profile ipv6-lookup** command to specify the IPv6 address lookup-mode.
   - To change from the default Host ID mode to the Network ID mode, enter **profile ipv6-lookup network-id**.

```
device(config-hardware)# profile ipv6-lookup network-id
```

   - To restore the default Host ID mode, enter **profile ipv6-lookup default**.

```
device(config-hardware)# profile ipv6-lookup default
```

UDAs

User-defined ACLs (UDAs) offer more flexibility for NPB than regular ACLs.

In NPB, there are flows that regular ACLs cannot filter. Packets in such flows require filtering based on deep packet inspection or on a combination of MAC and IP fields. UDAs—also known as Flex ACLs—parse deeper and more flexibly into packets for the needed filtering,

UDA implementation flow

A UDA starts functioning on an interface—for aggregation, replication, or forwarding—only if the following flow is implemented:

- Create a UDA profile, using the **uda-key profile** command.
- For the profile, specify the header types in the expected packet structure, using the **flow** command.
- For the profile, specify the header fields to match, using the **uda-key** command.
- Create a UDA, using the **uda access-list** command.
- Create one or more permit or deny rules in the UDA, using the [**seq** *seq-value*] { **deny** | **permit** } command.
- Create a route-map, using the **route-map** command.
- Apply the UDA to the route-map, using the **match uda address acl** command.
- In the route-map, specify the egress interface, using the **set interface** or **set next-hop-tvf-domain** command.
- Apply the profile to the interface, using the **uda-profile-apply** command.
- On a physical or port-channel interface, apply the route map to the ingress interface, using the **npb policy route-map** command.

Packet header-fields for UDAs

The following tables display the header fields supported for each packet header-type.

> NOTE
> For implementation details, refer to Configuring a UDA profile on page 21.

TABLE 5 ETHERNET (Ethernet header)

Field	Width (Bits)	Description
HAS_INNER_TAG	1	Packet has inner tag?

TABLE 5 ETHERNET (Ethernet header) (continued)

Field	Width (Bits)	Description
HAS_OUTER_TAG	1	Packet has outer tag?
HAS_8021BR_TAG	1	Packet has 802.1BR tag?
HAS_VNTAG	1	Packet has VN tag?
ETHER_TYPE	16	EtherType
INNER_VID	16	PCP or DEI or VLAN ID
OUTER_VID	16	PCP or DEI or VLAN ID
8021BR_TAG	32	802.1BR tag
VNTAG	32	VN tag
MAC_SA_16_47	32	Bits 16–47 of SA MAC
MAC_SA_0_15	16	Bits 0–15 of SA MAC
MAC_DA_32_47	16	Bits 32–47 of DA MAC
MAC_DA_0_31	32	Bits 0–31 of DA MAC

TABLE 6 TUN_ETHERNET (Inner Ethernet header in tunneled frames)

Field	Width (Bits)	Description
HAS_INNER_TAG	1	Packet has inner tag?
HAS_OUTER_TAG	1	Packet has outer tag?
ETHER_TYPE	16	EtherType
INNER_VID	16	PCP or DEI or VLAN ID
OUTER_VID	16	PCP or DEI or VLAN ID
MAC_SA_16_47	32	Bits 16–47 of SA MAC
MAC_SA_0_15	16	Bits 0–15 of SA MAC
MAC_DA_32_47	16	Bits 32–47 of DA MAC
MAC_DA_0_31	32	Bits 0–31 of DA MAC

TABLE 7 IPV4 (IPv4 header)

Field	Width (Bits)	Description
DIP	32	Destination IP
SIP	32	Source IP
PROTOCOL	8	IP protocol
TOTAL_LENGTH	16	Total length
TOS	8	Type of service (DSCP & ECN)
ECN	2	ECN
DSCP	6	DSCP

TABLE 8 IPV6 (IPv6 header)

Field	Width (Bits)	Description
NEXT_HEADER	8	Next header
TOTAL_LENGTH	16	Total length
TRAFFIC_CLASS	8	Traffic class (DSCP & ECN)
ECN	2	ECN

TABLE 8 IPV6 (IPv6 header) (continued)

Field	Width (Bits)	Description
DSCP	6	DSCP
DIP1	32	Bits 32-63 or 96-127 of Destination IP
DIP0	32	Bits 0-31 or 64-95 of Destination IP
SIP1	32	Bits 32-63 or 96-127 of Source IP
SIP0	32	Bits 0-31 or 64-95 of Source IP

TABLE 9 ARP (ARP header)

Field	Width (Bits)	Description
TARGET_IP	32	Target IP
TARGET_HW_ADDR_16_47	32	Bits 16-47 of target HW address
TARGET_HW_ADDR_0_15	16	Bits 0-15 of target HW address
SENDER_IP_16_31	16	Bits 16-31 of sender IP
SENDER_IP_0_15	16	Bits 0-15 of sender IP
SENDER_HW_ADDR_32_47	16	Bits 32-47 of sender HW address
SENDER_HW_ADDR_0_31	32	Bits 0-31 of sender HW address

TABLE 10 TCP (TCP header)

Field	Width (Bits)	Description
FLAGS	8	TCP flags
PORTS	32	Source and destination port
DST_PORT	16	Destination port
SRC_PORT	16	Source port

TABLE 11 UDP (UDP header)

Field	Width (Bits)	Description
PORTS	32	Source and destination port
DST_PORT	16	Destination port
SRC_PORT	16	Source port

TABLE 12 SCTP (SCTP header)

Field	Width (Bits)	Description
PORTS	32	Source and destination port
DST_PORT	16	Destination port
SRC_PORT	16	Source port

TABLE 13 IGMP (IGMP header)

Field	Width (Bits)	Description
TYPE	8	Type

TABLE 14 ICMP (ICMP header)

Field	Width (Bits)	Description
TYPE_CODE	16	ICMP type and code

TABLE 15 ICMPV6 (ICMPv6 header)

Field	Width (Bits)	Description
TYPE_CODE	16	ICMPv6 type and code

TABLE 16 MPLS (MPLS header)

Field	Width (Bits)	Description
LABEL0	24	Label 0 (Label, EXP & S-Bit)
LABEL1	24	Label 1 (Label, EXP & S-Bit)
LABEL2	24	Label 2 (Label, EXP & S-Bit)
LABEL3	24	Label 3 (Label, EXP & S-Bit)

TABLE 17 GRE (GRE header)

Field	Width (Bits)	Description
IS_NVGRE	1	Packet has NVGRE encapsulation?
IS_ERSPAN	1	Packet has ERSPAN encapsulation?
IS_L2	1	Is payload L2?
IS_IPV4	1	Is payload IPv4?
IS_IPV6	1	Is payload IPv4?
FLAGS	8	First byte of the GRE header
KEY_24_31	8	Bits 24–31 of GRE key
KEY_0_23	24	Bits 0–23 of GRE key
TNI	24	NVGRE tenant network ID

TABLE 18 VXLAN (VXLAN header)

Field	Width (Bits)	Description
VNI	24	VXLAN network identifier

TABLE 19 GTP (GTP header)

Field	Width (Bits)	Description
TEID	32	GTP tunnel endpoint ID

TABLE 20 PAYLOAD4 (4 bytes)

Field	Width (Bits)	Description
WORD0	32	Word 0 in the payload

TABLE 21 PAYLOAD8 (8 bytes)

Field	Width (Bits)	Description
WORD0	32	Word 0 in the payload

TABLE 21 PAYLOAD8 (8 bytes) (continued)

Field	Width (Bits)	Description
WORD1	32	Word 1 in the payload

TABLE 22 PAYLOAD16 (16 bytes)

Field	Width (Bits)	Description
WORD0	32	Word 0 in the payload
WORD1	32	Word 1 in the payload
WORD2	32	Word 2 in the payload
WORD3	32	Word 3 in the payload

TABLE 23 PAYLOAD32 (32 bytes)

Field	Width (Bits)	Description
WORD0	32	Word 0 in the payload
WORD1	32	Word 1 in the payload
WORD2	32	Word 2 in the payload
WORD3	32	Word 3 in the payload
WORD4	32	Word 4 in the payload
WORD5	32	Word 5 in the payload
WORD6	32	Word 6 in the payload
WORD7	32	Word 7 in the payload

Headers larger than 128 bytes

If a frame exceeds the 128-byte limit, parsing is limited to the header elements under the limit.

Consider a frame with format ETH > IPv6 > UDP > GTP > IPv6 > TCP > payload. Depending on which tags are in the ETH header, such a frame varies from 130 through 150 bytes.

If the 128-byte parsing limit is exceeded, the frame is parsed as ETH > IPv6 > UDP > GTP > IPv6 instead of ETH > IPv6 > UDP > GTP > IPv6 > TCP.

Configuring a UDA profile

Use this task to create a UDA profile and define its directives.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **uda-key profile** command to create a UDA profile.

```
device(config)# uda-key profile prof_01
```

3. Enter the **flow** command to define the header types of the expected packet structure.

```
device(conf-uda-profile-prof_01)# flow header0 ETHERNET header1 IPV4 header2 UDP header3 PAYLOAD32
```

4. Enter the **uda-key** commands to assign header fields to UDA keys.

```
device(conf-uda-profile-prof_01)# uda-key0 header0 ETHER_TYPE
device(conf-uda-profile-prof_01)# uda-key1 header1 PROTOCOL
device(conf-uda-profile-prof_01)# uda-key2 header2 DST_PORT
device(conf-uda-profile-prof_01)# uda-key3 header3 WORD1
```

The following example creates a UDA profile and defines its directives.

```
device# configure terminal
device(config)# uda-key profile prof_01
device(conf-uda-profile-prof_01)# flow header0 ETHERNET header1 IPV4 header2 UDP header3 PAYLOAD32
device(conf-uda-profile-prof_01)# uda-key0 header0 ETHER_TYPE
device(conf-uda-profile-prof_01)# uda-key1 header1 PROTOCOL
device(conf-uda-profile-prof_01)# uda-key2 header2 DST_PORT
device(conf-uda-profile-prof_01)# uda-key3 header3 WORD1
```

Applying a UDA profile on a physical interface

Use this task to apply a user-defined ACL (UDA)-profile on an Ethernet interface.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command—specifying the slot and port—to access interface configuration mode.

```
device(config)# interface ethernet 0/2
```

3. Enter the **uda-profile-apply** command—specifying the UDA profile—to apply the UDA profile to the interface.

```
device(conf-if-eth-0/2)# uda-profile-apply prof_01
```

Applying a UDA profile on a port-channel interface

Use this task to apply a user-defined ACL (UDA)-profile on a port-channel interface.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface port-channel** command, specifying the port-channel number.

```
device(config)# interface port-channel 10
```

3. Enter the **uda-profile-apply** command, specifying the UDA profile.

```
device(config-Port-channel-10)# uda-profile-apply prof_02
```

Creating a UDA

Use this task to create a user-defined ACL (UDA) and define permit and deny rules within.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **uda access-list** command to create the access list.

```
device(config)# uda access-list extended uda_01
```

3. Enter rules, specifying the needed parameters.

```
device(conf-uda-acl-ext)# permit 0x0800 0xFFFF 17 0xFF 3400 0xFFFF 0x11223344 0xFFFFFFFF
```

The following example creates a UDA and defines a permit rule, with statistics enabled for the rule.

```
device# configure terminal
device(config)# uda access-list extended uda_01
device(conf-uda-acl-ext)# permit 0x112233 0xFFFFFF 0x0a0a0001 0xFFFFFFFF 0x0a0b0001 0xFFFFFFFF 0x11223344
0xFFFFFFFF
```

UDA examples

The following examples illustrate the interplay between UDA profile-definitions and UDA rules.

Header flow: ETH > IPv4 > UDP > RAW payload

Matches:

- IPv4-UDP DPORT = 3400
- Second word (4 bytes) of the payload (following UDP)

```
device# configure terminal
device(config)# uda-key profile fkp1
device(conf-uda-profile-fkp1)# flow header0 ETHERNET header1 IPV4 header2 UDP header3 PAYLOAD32
device(conf-uda-profile-fkp1)# uda-key0 header0 ETHER_TYPE
device(conf-uda-profile-fkp1)# uda-key1 header1 PROTOCOL
device(conf-uda-profile-fkp1)# uda-key2 header2 DST_PORT
device(conf-uda-profile-fkp1)# uda-key3 header3 WORD1
device(conf-if-eth-0/1)# uda-profile-apply fkp1
device(config)# uda access-list extended uda_acl_1
device(conf-uda-acl-ext)# permit 0x0800 0xFFFF 0x11 0xFF 0xD48 0xFFFF 0x11223344 0xFFFFFFFF
```

Header flow: ETH > IPv4 > UDP > VXLAN

Match: VXLAN VNID

```
device# configure terminal
device(config)# uda-key profile fkp1
device(conf-uda-profile-fkp1)# flow header0 ETHERNET header1 IPV4 header2 UDP header3 VXLAN
device(conf-uda-profile-fkp1)# uda-key0 header0 ETHER_TYPE
device(conf-uda-profile-fkp1)# uda-key1 header1 PROTOCOL
device(conf-uda-profile-fkp1)# uda-key2 header2 DST_PORT
device(conf-uda-profile-fkp1)# uda-key3 header3 VNI
device(conf-if-eth-0/1)# uda-profile-apply fkp1
device(config)# uda access-list extended uda_acl_2
device(conf-uda-acl-ext)# permit 0x0800 0xFFFF 0x11 0xFF 0x12B5 0xFFFF 0x1F4 0xffffff
```

Header flow: ETH > IPv6 > GRE > IPv4 > TCP > RAW payload (eHRPD)

Match: TCP DPORT = 443

```
device# configure terminal
device(config)# uda-key profile fkp2
device(conf-uda-profile-fkp2)# flow header0 ETHERNET header1 IPV6 header2 GRE header3 IPV4 header4 TCP
device(conf-uda-profile-fkp2)# uda-key0 header4 DST_PORT
device(conf-if-eth-0/2)# uda-profile-apply fkp2
```

```
device(config)# uda access-list extended uda_acl_3
device(conf-uda-acl-ext)# permit 0x1BB 0xFFFF 0 0 0 0 0 0
```

ETH > IPv4 > UDP > VxLAN > ETH > IPv4 > TCP > RAW Payload (VXLAN with IPv4 passenger)

Matches:

- VNI
- Inner IPv4 SIP and DIP

```
device# configure terminal
device(config)# uda-key profile fkp3
device(conf-uda-profile-fkp3)# flow header0 ETHERNET header1 IPV4 header2 UDP header3 VXLAN header4
TUN_ETHERNET header5 IPV4
device(conf-uda-profile-fkp3)# uda-key0 header3 VNI
device(conf-uda-profile-fkp3)# uda-key1 header5 SIP
device(conf-uda-profile-fkp3)# uda-key2 header5 DIP
device(conf-if-eth-0/3)# uda-profile-apply fkp3
device(config)# uda access-list extended uda_acl_4
device(conf-uda-acl-ext)# permit 0x112233 0xFFFFFF 0x0a0a0001 0xFFFFFFFF 0x0a0b0001 0xFFFFFFFF 0 0
```

Traffic Aggregation and Replication

Aggregating traffic from interfaces

This task aggregates and forwards traffic from multiple interfaces to a single, physical egress interface.

1. Configure a regular ACL or a user-defined ACL (UDA):
   - For a regular ACL, refer to Creating ACLs for NPB on page 16.

     ```
     device(config)# ip access-list standard aclNPB_01
     device(conf-ipacl-std)# permit host 192.1.1.1 count
     device(conf-ipacl-std)# exit
     ```

   - For a UDA, refer to UDAs on page 17.
2. Configure a route map that contains the relevant **match { mac | ip | ipv6 | uda } address acl** command.

   ```
   device(config)# route-map npb_map1 permit 1
   device(config-route-map-npb_map1/permit/1)# match ip address acl acl_2
   ```

3. Specify the route-map egress physical interface.
   - Without stripping 802.1q VLAN tags:

     ```
     device(config-route-map-npb_map1/permit/1)# set interface ethernet 0/5
     ```

   - Stripping 802.1q VLAN tags:

     ```
     device(config-route-map-npb_map1/permit/1)# set interface ethernet 0/5 strip-vlan outer
     ```

   - Adding a 802.1q VLAN tag (optionally, in addition to **strip-vlan outer**):

     ```
     device(config-route-map-npb_map1/permit/1)# set interface ethernet 0/5 add-vlan outer 2
     ```

4. Exit to global configuration mode.

   ```
   device(config-route-map-npb_map1/permit/1)# exit
   ```

5. Apply the route map to the ingress interfaces.
   - Physical interfaces

     ```
     device(config)# interface ethernet 0/2
     device(conf-if-eth-0/2)# npb policy route-map npb_map1
     ```

   - Port-channel interfaces

     ```
     device(config)# interface port-channel 10
     device(config-Port-channel-10)# npb policy route-map npb_map1
     ```

The following example configures ingress traffic from Ethernet 0/1 and port-channel 100 to egress Ethernet 0/5.

```
device# configure terminal
device(config)# route-map npb_map permit 10
device(config-route-map-npb_map/permit/10)# match ip address acl acl_2
device(config-route-map-npb_map/permit/10)# set interface ethernet 0/5
device(config-route-map-npb_map/permit/10)# exit
device(config)# interface ethernet 0/1
device(conf-if-eth-0/1)# npb policy route-map npb_map
device(conf-if-eth-0/1)# exit
device(config)# interface port-channel 100
device(config-Port-channel-100)# npb policy route-map npb_map
```

Replication to multiple interfaces

Traffic on an ingress interface is replicated and forwarded on multiple egress interfaces, usually to multiple monitoring tools.

The replication ingress-interface is one physical or port-channel interface.

Multiple physical and port-channel egress interfaces are supported. However, you first need to associate such interfaces with a Transparent VLAN flooding (TVF) domain. You then use a route map to designate that TVF domain as egress. Ingress traffic is replicated, and forwarded by way of the TVF to the egress interfaces that you specified.

Transparent VLAN flooding (TVF)

TVF forwards packets without any form of CPU intervention, including MAC learning and MAC destination lookups.

This implementation of TVF has the following attributes:

- Traffic is distributed in hardware to all members of the TVF domain.
- Because this feature does not use any MAC address entries in the CAM, it is useful when MAC address entries need to be conserved.
- You can create as many as 4096 TVF domains.
- Domain members can be tagged and untagged ports. There is no software limitation on the number of member ports.
- You can mix and match ports with different speeds.
- At the TVF domain level, load balancing does not occur. If you need load balancing, associate a port-channel with the TVF domain, which balances the loads among the interfaces included in the port-channel.
- CPU intervention is not required, enabling line-rate traffic forwarding.

Creating TVF domains

This task creates one or more Transparent VLAN flooding (TVF) domains.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **tvf-domain** command, specifying one of the valid formats.
   - To create one TVF domain, specify an integer from 1 through 4096.

     ```
     device(config)# tvf-domain 10
     ```

   - To specify a range of TVF domains, insert a hyphen (-) between the beginning and ending integers.

     ```
     device(config)# tvf-domain 20-30
     ```

   - To specify individual domains and ranges of domains, separate them with commas (for example, 1,5-7,55).

     ```
     device(config)# tvf-domain 1,5-7,55
     ```

3. To add a description of a TVF domain, enter the **description** command.

   ```
   device(config)# tvf-domain 10
   device(config-tvf-domain-10)# description Sample TVF description
   ```

4. Enter **exit** to return to global configuration mode.

   ```
   device(config-tvf-domain-10)# exit
   device(config)#
   ```

Assigning a TVF domain to a physical egress interface

This task assigns a TVF domain to a physical egress interface.

1. Enter **configure terminal** to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **interface ethernet** command to access interface configuration mode.

   ```
   device(config)# interface ethernet 0/5
   ```

3. Enter the **tvf-domain** command, specifying one of the valid formats.
   - To assign one TVF domain to the interface, specify its integer ID.

     ```
     device(conf-if-eth-0/5)# tvf-domain add 10
     ```

   - To assign a range of TVF domains to the interface, insert a hyphen (-) between the beginning and ending integers.

     ```
     device(conf-if-eth-0/5)# tvf-domain add 20-30
     ```

   - To assign individual domains and ranges of domains, separate them with commas (for example, 1,5-7,55).

     ```
     device(conf-if-eth-0/5)# tvf-domain add 1,5-7,55
     ```

   - To assign all defined TVF domains to the interface, enter **tvf-domain all**.

     ```
     device(conf-if-eth-0/5)# tvf-domain all
     ```

   - To assign all defined TVF domains to the interface—except for those specified—enter the **tvf-domain except** option.

     ```
     device(conf-if-eth-0/5)# tvf-domain except 1,2,4-7
     ```

Assigning a TVF domain to a port-channel egress interface

This task assigns a TVF domain to a port-channel egress interface.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface port-channel** command to access port-channel configuration mode.

```
device(config)# interface port-channel 10
```

3. Enter the **tvf-domain** command, specifying one of the valid formats.
   - To assign one TVF domain to the interface, specify its integer ID.

```
device(config-Port-channel-10)# tvf-domain add 10
```

   - To assign a range of TVF domains to the interface, insert a hyphen (-) between the beginning and ending integers.

```
device(config-Port-channel-10)# tvf-domain add 20-30
```

   - To assign individual domains and ranges of domains, separate them with commas (for example, 1,5-7,55).

```
device(config-Port-channel-10)# tvf-domain add 1,5-7,55
```

   - To assign all defined TVF domains to the interface, enter **tvf-domain all**.

```
device(config-Port-channel-10)# tvf-domain all
```

   - To assign all defined TVF domains to the interface—except for those specified—enter the **tvf-domain except** option.

```
device(config-Port-channel-10)# tvf-domain except 1,2,4-7
```

Replicating traffic to multiple interfaces

This task replicates traffic entering an interface to multiple egress interfaces.

1. Create needed TVF domains, as described in Creating TVF domains on page 26.

```
device# configure terminal
device(config)# tvf-domain 10
device(config-tvf-domain-10)# exit
```

2. Assign TVF domains to the egress interfaces.
   - Physical interfaces (For details, refer to Assigning a TVF domain to a physical egress interface on page 27.)

```
device(config)# interface ethernet 0/5
device(conf-if-eth-0/5)# tvf-domain add 10
```

   - Port-channel interfaces

```
device(config)# interface port-channel 10
device(config-Port-channel-10)# tvf-domain add 10
```

3. Configure a regular ACL or a user-defined ACL (UDA):
   - For a regular ACL, refer to Creating ACLs for NPB on page 16.

     ```
     device(config)# ip access-list standard aclNPB_01
     device(conf-ipacl-std)# permit host 192.1.1.1 count
     device(conf-ipacl-std)# exit
     ```

   - For a UDA, refer to UDAs on page 17.
4. Configure a route map that contains the relevant **match { mac | ip | ipv6 | uda } address acl** command.

   ```
   device(config)# route-map npb_map1 permit 1
   device(config-route-map-npb_map1/permit/1)# match ip address acl acl_2
   ```

5. Specify the TVF domain that contains the egress interfaces for the replicated traffic.
   - Without stripping 802.1q VLAN tags:

     ```
     device(config-route-map-npb_map1/permit/1)# set next-hop-tvf-domain 5
     ```

   - Stripping 802.1q VLAN tags:

     ```
     device(config-route-map-npb_map1/permit/1)# set next-hop-tvf-domain 5 strip-vlan outer
     ```

   - Adding a 802.1q VLAN tag (optionally, in addition to **strip-vlan outer**):

     ```
     device(config-route-map-npb_map1/permit/1)# set next-hop-tvf-domain 5 add-vlan outer 2
     ```

6. Exit to global configuration mode.

   ```
   device(config-route-map-npb_map1/permit/1)# exit
   ```

7. Apply the route map to the ingress interface.
   - Physical interface

     ```
     device(config)# interface ethernet 0/2
     device(conf-if-eth-0/2)# npb policy route-map npb_map1
     ```

   - Port-channel interface

     ```
     device(config)# interface port-channel 10
     device(config-Port-channel-10)# npb policy route-map npb_map1
     ```

The following example replicates traffic entering an interface to multiple egress interfaces.

```
device# configure terminal
device(config)# tvf-domain 10
device(config-tvf-domain-10)# description Sample TVF description
device(config-tvf-domain-10)# exit
device(config)# interface ethernet 0/5
device(conf-if-eth-0/5)# tvf-domain add 10
device(conf-if-eth-0/5)# exit
device(config)# interface port-channel 10
device(config-Port-channel-10)# tvf-domain add 10
device(config-Port-channel-10)# exit
device(config)# route-map npb_map1 permit 1
device(config-route-map-npb_map1/permit/1)# match ip address acl acl_2
device(config-route-map-npb_map1/permit/1)# set next-hop-tvf-domain 5
device(config-route-map-npb_map1/permit/1)# exit
device(config)# interface ethernet 0/2
device(conf-if-eth-0/2)# npb policy route-map npb_map1
```

Load Balancing Under NPB

Load balancing overview

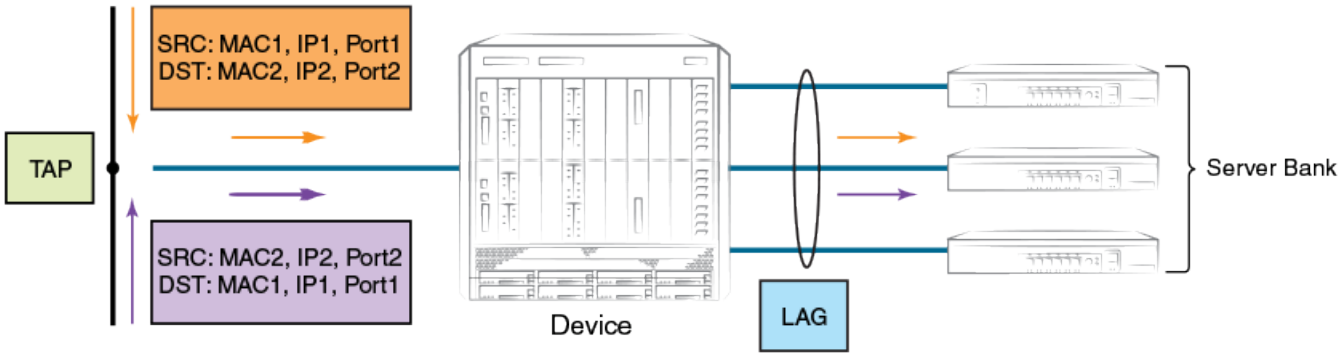
The only type of link-aggregation load-balancing supported under Network Packet Broker (NPB) is symmetric load balancing.

NOTE
For details on link aggregation, including link-aggregation groups (LAGs), refer to the "Link Aggregation" section of the *Extreme SLX-OS Layer 2 Switching Configuration Guide*. LAGs are also referred to as *port-channels*.

Symmetric load-balancing interchanges source and destination addresses to ensure that bidirectional traffic between a source and destination pair flows through one LAG member. Under NPB, symmetric load-balancing is enabled by default and cannot be disabled. However, you can modify the load-balancing parameters.

For network telemetry applications, network traffic is tapped and sent to a device that hashes selected traffic to the application servers downstream. For many monitoring and security applications, bidirectional conversations flowing through the system must be carried on the same port of a port-channel (LAG). In addition, firewalls between devices can be configured to allow such bidirectional conversations.

FIGURE 1 Symmetric load balancing



Load balancing of tunneled frames, with header stripping

Load balancing of tunneled frames (VXLAN, NVGRE, ERSPAN, GTP, or MPLS pseudo-wire) is affected by header stripping:

- Without header stripping, load balancing is based on the hash produced from outer headers.
- With header stripping, load balancing is based on the hash produced from inner headers.

NOTE
For details of header stripping, refer to Interface-level header stripping on page 38.

Configuring symmetric load balancing

Use this task to change the load-balancing mode from the default **src-dst-ip-mac-vid-port** mode or another current mode to the load-balancing mode that you require.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **load-balance** command to specify the mode that you require.

```
device(conf)# load-balance src-dst-ip
```

Symmetric load-balancing options and examples

The following tables display examples of the NPB symmetric load-balancing hashing options.

src-dst-ip

The following table displays inputs for a **load-balance src-dst-ip** example. The distribution is based on source and destination IPv4 or IPv6 addresses.

TABLE 24 Symmetric load-balancing **src-dst-ip** option

Flow	Source IP	Destination IP	Source MAC	Destination MAC	VLAN
1	IPA	IPB	MACA	MACB	—
2	IPB	IPA	MACD	MACE	—
3	IPD	IPE	MACA	MACB	VID2
4	IPA	IPC	MACA	MACD	VID2

Because flows 1 and 2 share the same source and destination IP addresses (the other fields are not considered), they are load balanced on the same port-channel port.

src-dst-ip-mac-vid

The following table displays inputs for a **load-balance src-dst-ip-mac-vid** example. The distribution is based on source and destination IPv4 or IPv6 and MAC addresses; and outer VLAN ID (VID).

TABLE 25 Symmetric load-balancing **src-dst-ip-mac-vid** option

Flow	Source IP	Destination IP	Source MAC	Destination MAC	VLAN
1	IPA	IPB	MACA	MACB	VID1
2	IPB	IPA	MACB	MACA	VID1
3	IPA	IPB	MACA	MACB	VID2

TABLE 25 Symmetric load-balancing **src-dst-ip-mac-vid** option (continued)

Flow	Source IP	Destination IP	Source MAC	Destination MAC	VLAN
4	IPA	IPB	MACD	MACE	VID2

Because flows 1 and 2 share the same source and destination IP and MAC addresses and the same VLAN, they are load-balanced on the same port-channel port.

src-dst-ip-mac-vid-port

The following table displays inputs for a **load-balance src-dst-ip-mac-vid-port** example. The distribution is based on source and destination IPv4 or IPv6 and MAC addresses, VID, and TCP or UDP destination port.

NOTE
This is the default **load-balance** option.

TABLE 26 Symmetric load-balancing **src-dst-ip-mac-vid-port** option

Flow	Source IP	Destination IP	Source MAC	Destination MAC	Source L4 Port	Destination L4 Port	VLAN
1	IPA	IPB	MACA	MACB	P1	P2	VID1
2	IPB	IPA	MACB	MACA	P2	P1	VID1
3	IPA	IPB	MACA	MACB	P2	P1	VID2
4	IPA	IPB	MACD	MACE	P3	P4	VID2

Because flows 1 and 2 share the same source and destination IP and MAC addresses, the same TCP or UDP ports, and the same VLAN, they are load-balanced on the same port-channel port.

src-dst-ip-port

The following table displays inputs for a **load-balance src-dst-ip-port** example. The distribution is based on source and destination IPv4 or IPv6 address and TCP or UDP destination port.

TABLE 27 Symmetric load-balancing **src-dst-ip-port** option

Flow	Source IP	Destination IP	Source L4 Port	Destination L4 Port
1	IPA	IPB	P1	P2
2	IPB	IPA	P2	P1
3	IPA	IPB	P2	P1
4	IPA	IPB	P3	P4

Because flows 1 and 2 share the same source and destination IP addresses and the same TCP or UDP ports, they are load-balanced on the same port-channel port.

src-dst-mac-vid

The following table displays inputs for a **load-balance src-dst-mac-vid** example. The distribution is based on the source and destination MAC addresses and VID.

TABLE 28 Symmetric load-balancing **src-dst-mac-vid** option

Flow	Source MAC	Destination MAC	VLAN
1	MACA	MACB	VID1
2	MACB	MACA	VID1
3	MACA	MACB	VID2
4	MACD	MACE	VID2

Because flows 1 and 2 share the same source and destination MAC addresses and the same VLAN, they are load-balanced on the same port-channel port.

Forwarding traffic to a port-channel

For load balancing, this task forwards traffic entering an interface to a port-channel.

1. Configure a regular ACL or a user-defined ACL (UDA):
   - For a regular ACL, refer to Creating ACLs for NPB on page 16.

     ```
     device(config)# ip access-list standard aclNPB_01
     device(conf-ipacl-std)# permit host 192.1.1.1 count
     device(conf-ipacl-std)# exit
     ```

   - For a UDA, refer to UDAs on page 17.
2. Configure a route map that contains the relevant **match { mac | ip | ipv6 | uda } address acl** command.

   ```
   device(config)# route-map npb_map1 permit 1
   device(config-route-map-npb_map1/permit/1)# match ip address acl aclNPB_01
   ```

3. Specify the egress port-channel interface.
   - Without stripping 802.1q VLAN tags:

     ```
     device(config-route-map-npb_map1/permit/1)# set interface port-channel 10
     ```

   - Stripping 802.1q VLAN tags:

     ```
     device(config-route-map-npb_map1/permit/1)# set interface port-channel 10 strip-vlan outer
     ```

   - Adding a 802.1q VLAN tag (optionally, in addition to **strip-vlan outer**):

     ```
     device(config-route-map-npb_map1/permit/1)# set interface ethernet 0/5 add-vlan outer 2
     ```

4. Exit to global configuration mode.

```
device(config-route-map-npb_map1/permit/1)# exit
```

5. Apply the route map to a single ingress interface:
   - Physical interface

```
device(config)# interface ethernet 0/2
device(conf-if-eth-0/2)# npb policy route-map npb_map1
```

   - Port-channel interface

```
device(config)# port-channel 5
device(config-Port-channel-10)# npb policy route-map npb_map1
```

The following example forwards traffic entering a physical interface to a port-channel.

```
device# configure terminal
device(config)# ip access-list standard aclNPB_01
device(conf-ipacl-std)# permit host 192.1.1.1 count
device(conf-ipacl-std)# exit
device(config)# route-map npb_map1 permit 1
device(config-route-map-npb_map1/permit/1)# match ip address acl aclNPB_01
device(config-route-map-npb_map1/permit/1)# set interface port-channel 10
device(config-route-map-npb_map1/permit/1)# exit
device(config)# interface ethernet 0/2
device(conf-if-eth-0/2)# npb policy route-map npb_map1
```

Header Modification

Header-modification overview

Protocol headers help packets reach their destinations, but are not needed by the security and monitoring tools to which NPB forwards traffic.

Tagging and encapsulation techniques have long been a part of networking. However, the recent adoption of new encapsulation protocols—such as VXLAN, VN-Tag, and 802.1BR—can create visibility blind spots, because some visibility applications were not designed to interpret these new protocols.

By removing the encapsulation header, the NPB removes the burden of interpreting the various encapsulation protocols from the visibility applications. Therefore, the header-stripping feature enables network operators to deploy new encapsulation protocols without interfering with proper functioning of previously deployed visibility applications. Other header stripping benefits include:

- Reduced packet overhead and better visibility-application bandwidth utilization.
- Application of defined Layer 2 and Layer 3 ACLs to the inner headers.
- Load balancing based on the inner headers.

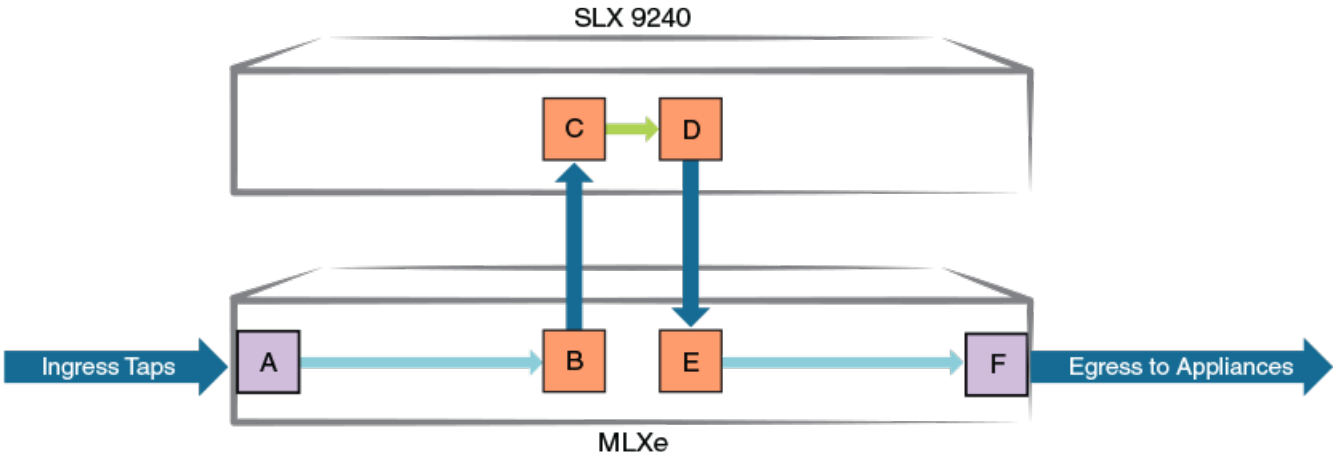
The following types of header modification are supported:

- Interface-level header stripping on page 38
- Configuring GTP-HTTPS frame filtering on page 48
- VLAN-header flow modification on page 49

Header-modification flow

The following figure displays a sample header-modification flow.

FIGURE 2 Header-modification flow



In the example flow, headers or frames of a specified type are removed, as follows:

1. Traffic enters an MLXe through port A.
2. The traffic exits the MLXe from port B and enters the SLX 9240 through port C. If a header-modification command is enabled on port C, specified headers or frames are removed from the flow.
3. Traffic is forwarded from SLX 9240 port C to port D.
4. Traffic is forwarded back to MLXe port E, using an NPB route-map. (For details, refer to Route maps under NPB on page 12.)
5. Traffic is forwarded from MLXe port E to port F, using an NPB route-map.
6. Traffic is forwarded from MLXe port F to monitoring tools.

Interface-level header stripping

Interface-level header stripping enables you to configure different types of header stripping on different interfaces.

Header-stripping configuration guidelines

You cannot configure 802.1BR and VN-Tag header-stripping on the same interface.

You can configure other combinations of header-stripping types on an interface. However, only one kind of stripping operation is actually implemented. For more detailed guidelines, refer to the "Configuration guidelines" sections for the various header types:

- 802.1BR header stripping on page 39
- VN-Tag header stripping on page 40
- VXLAN header stripping on page 42
- NVGRE header stripping on page 43

Header-stripping configuration guidelines (tunneled frames)

The following header-stripping configuration guidelines apply to tunneled frames (VXLAN, NVGRE, ERSPAN, GTP, or MPLS pseudo-wire):

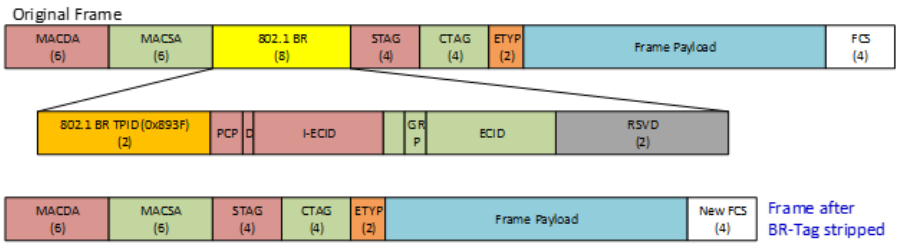
- Header stripping affects only the outer, encapsulation headers.
- Without header stripping, Layer 2 and Layer 3 ACLs act on outer headers; with header stripping, ACLs act on the inner headers.
- For load balancing configuration guidelines (including for tunneled frames with header stripping), refer to Load balancing of tunneled frames, with header stripping on page 31.

802.1BR header stripping

The following figure shows frame structure before and after 802.1BR header stripping.

Stripping the 802.1BR header prepares the header and frame payload for forwarding and processing.

FIGURE 3 Before and after 802.1BR header stripping



Configuration guidelines for 802.1BR header stripping

By default, interfaces support 802.1BR tags but not VN-Tags. For instructions how to toggle between the two header modes, refer to Configuring 802.1BR header stripping on page 40 and Configuring VN-Tag header stripping on page 41.

If a tunneled frame has an 802.1BR tag in the outer L2 header, VXLAN, NVGRE, ERSPAN, or L2-over-MPLS/pseudo-wire header-stripping also deletes the 802.1BR tag. (802.1BR tags in the inner L2 header are not supported.)

If both MPLS and 802.1BR header-stripping are configured, MPLS is always implemented. However, details vary between IP-over-MPLS and L2-over-MPLS regarding 802.1BR tags:

- Under IP-over-MPLS, only the MPLS labels are stripped; 802.1BR tags are not affected. The header diagram for this case is as follows:

IP-over-MPLS frame headers			
L2 (802.1BR tag)	MPLS	IPv4	TCP

- Under L2-over-MPLS (pseudo-wire), both the outer L2 (with 802.1BR tag) and the MPLS header are stripped. The header diagram for this case is as follows:

L2-over-MPLS frame (pseudo-wire) headers				
L2 (802.1BR tag)	MPLS	L2	IPv4	TCP

Configuring 802.1BR header stripping

Use this task to enable or disable 802.1BR header stripping on an Ethernet interface.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command to access Ethernet interface configuration mode.

```
device(config)# interface ethernet 0/2
```

3. If default support for 802.1BR headers was changed on that interface to support for VN-Tag headers, enter **no allow-vn-tag** to restore support for 802.1BR headers.

```
device(conf-if-eth-0/2)# no allow-vn-tag
```

4. Enable or disable this feature on the interface.
   - To enable 802.1BR header stripping, enter **strip-802-1br**.

```
device(conf-if-eth-0/2)# strip-802-1br
```

   - To disable 802.1BR header stripping, enter **no strip-802-1br**.

```
device(conf-if-eth-0/2)# no strip-802-1br
```

The following example enables 802.1BR header stripping on an interface.

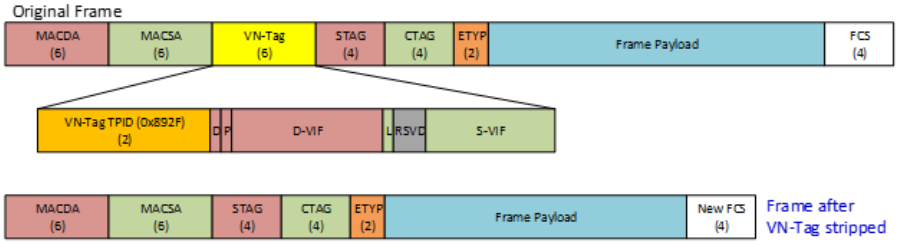
```
device# configure terminal
device(config)# interface ethernet 0/2
device(conf-if-eth-/2)# strip-802-1br
```

VN-Tag header stripping

The following figure shows frame structure before and after 802.1BR header stripping.

Stripping the VN-Tag prepares the header and frame payload for forwarding and processing.

FIGURE 4 Before and after VN-Tag header stripping



Configuration guidelines for VN-Tag header stripping

By default, interfaces support 802.1BR tags but not VN-Tags. For instructions how to toggle between the two header modes, refer to Configuring 802.1BR header stripping on page 40 and Configuring VN-Tag header stripping on page 41.

If a tunneled frame has a VN-Tag in the outer L2 header, VXLAN, NVGRE, ERSPAN, or L2-over-MPLS/pseudo-wire header-stripping also deletes the VN-Tag. (VN-Tags in the inner L2 header are not supported.)

If both MPLS and VN-Tag header-stripping are configured, MPLS is always implemented. However, details vary between IP-over-MPLS and L2-over-MPLS regarding VN-Tags:

- Under IP-over-MPLS, only the MPLS labels are stripped; VN-Tags are not affected. The header diagram for this case is as follows:

IP-over-MPLS frame headers			
L2 (VN-Tag)	MPLS	IPv4	TCP

- Under L2-over-MPLS (pseudo-wire), both the outer L2 (with VN-Tag) and the MPLS header are stripped. The header diagram for this case is as follows:

IP-over-MPLS frame (pseudo-wire) headers				
L2 (VN-Tag)	MPLS	L2	IPv4	TCP

Configuring VN-Tag header stripping

Use this task to enable or disable Virtual NIC (VN)-Tag header stripping on an Ethernet interface.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command to access Ethernet interface configuration mode.

```
device(config)# interface ethernet 0/2
```

3. If default support for 802.1BR headers is in effect on that interface, enter **allow-vn-tag** to support for VN-Tag headers.

```
device(conf-if-eth-0/2)# allow-vn-tag
```

4. Enable or disable this feature on the interface.
   - To enable VN-Tag header stripping, enter **strip-vn-tag**.

```
device(conf-if-eth-0/2)# strip-vn-tag
```

   - To disable VN-Tag header stripping, enter **no strip-vn-tag**.

```
device(conf-if-eth-0/2)# no strip-vn-tag
```

The following example enables VN-Tag header stripping on an interface.

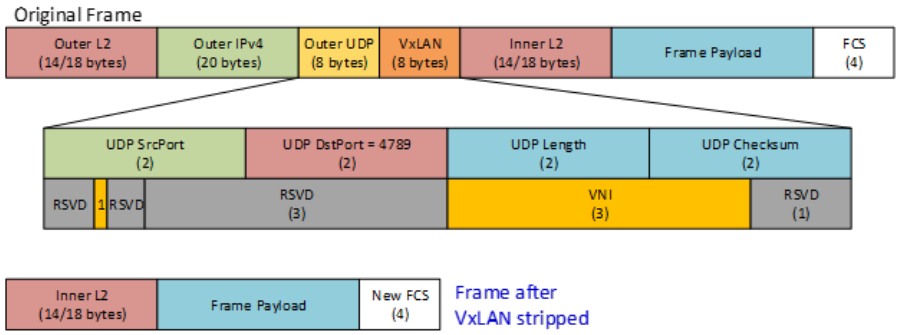
```
device# configure terminal
device(config)# interface ethernet 0/2
device(conf-if-eth-0/2)# strip-vn-tag
```

VXLAN header stripping

The following figure shows frame structure before and after VXLAN header-stripping.

VXLAN-encapsulated frames have an outer L2-IPv4-UDP-VXLAN header structure. Stripping the outer headers prepares the inner headers and frame payload for forwarding and processing.

FIGURE 5 Before and after VXLAN header stripping



Configuration guidelines for VXLAN header stripping

For tunneled frames, header-stripping affects only the outer, encapsulation headers. For example:

- If both ERSPAN and VXLAN header-stripping are configured on the ingress interface, only the ERSPAN headers are stripped. The header diagram for this case is as follows:

ERSPAN headers				VXLAN headers				Payload frame headers		
L2	IPv4	GRE	ERSPAN	L2	iIPv4	UPD	VXLAN	L2	IPv4	TCP

- If both VXLAN and MPLS header-stripping are configured, only VXLAN headers are stripped. The header diagram for this case is as follows:

VXLAN headers				Payload frame headers			
L2	IPv4	UDP	VXLAN	L2	MPLS	IPv4	TCP

Configuring VXLAN header stripping

Use this task to enable or disable Virtual Extensible LAN (VXLAN) header stripping on an Ethernet interface.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command to access Ethernet interface configuration mode.

```
device(config)# interface ethernet 0/2
```

3. Enable or disable this feature on the interface.
   - To enable VXLAN header stripping, enter **strip-vxlan**.

     ```
     device(conf-if-eth-0/2)# strip-vxlan
     ```

   - To disable VXLAN header stripping, enter **no strip-vxlan**.

     ```
     device(conf-if-eth-0/2)# no strip-vxlan
     ```

The following example enables VXLAN header stripping on an interface.

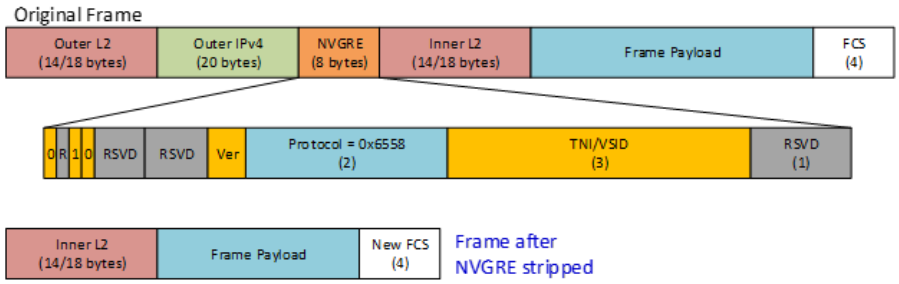
```
device# configure terminal
device(config)# interface ethernet 0/2
device(conf-if-eth-0/2)# strip-vxlan
```

NVGRE header stripping

The following figure shows frame structure before and after NVGRE header stripping.

NVGRE-encapsulated frames have an outer L2-IPv4-NVGRE header structure. Stripping the outer headers prepares the inner headers and frame payload for forwarding and processing.

FIGURE 6 Before and after NVGRE header stripping



Configuration guidelines for NVGRE header stripping

For tunneled frames, header-stripping affects only the outer, encapsulation headers. For example:

- If both ERSPAN and NVGRE header-stripping are configured on the ingress interface, only the ERSPAN headers are stripped. (NVGRE header-stripping alone has no effect.) The header diagram for this case is as follows:

ERSPAN headers				NVGRE headers			Payload frame headers		
L2	IPv4	GRE	ERSPAN	L2	IPv4	NVGRE	L2	IPv4	TCP

- If both NVGRE and MPLS header-stripping are configured, only NVGRE headers are stripped; MPLS labels are left untouched. The header diagram for this case is as follows:

NVGRE headers			Payload frame headers			
L2	IPv4	NVGRE	L2	MPLS	IPv4	TCP

Configuring NVGRE header stripping

Use this task to enable or disable Network Virtualization using Generic Routing Encapsulation (NVGRE) header stripping on an Ethernet interface.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command to access Ethernet interface configuration mode.

```
device(config)# interface ethernet 0/2
```

3. Enable or disable this feature on the interface.
   - To enable NVGRE header stripping, enter **strip-nvgre**.

```
device(conf-if-eth-0/2)# strip-nvgre
```

   - To disable NVGRE header stripping, enter **no strip-nvgre**.

```
device(conf-if-eth-0/2)# no strip-nvgre
```

The following example enables NVGRE header stripping on an interface.

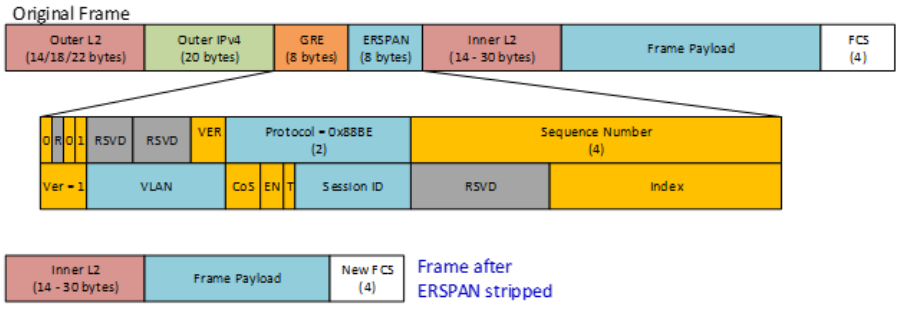
```
device# configure terminal
device(config)# interface ethernet 0/2
device(conf-if-eth-0/2)# strip-nvgre
```

ERSPAN-II header-stripping

The following figure shows frame structure before and after ERSPAN Type II header stripping.

ERSPAN-II-encapsulated frames have an outer L2-IPv4-GRE-ERSPAN header structure. Stripping the outer headers prepares the inner headers and frame payload for forwarding and processing.

FIGURE 7 Before and after ERSPAN-II header-stripping



Configuration guidelines for ERSPAN-II header-stripping

For tunneled frames, header-stripping affects only the outer, encapsulation headers. For example:

- If both ERSPAN and VXLAN header-stripping are configured on the ingress interface, only the ERSPAN headers are stripped. (VXLAN header-stripping alone has no effect.) The header diagram for this case is as follows:

ERSPAN headers				VXLAN headers				Payload frame headers		
L2	IPv4	GRE	ERSPAN	L2	IPv4	GRE	ERSPAN	L2	IPv4	TCP

- If both ERSPAN and MPLS header-stripping are configured, only ERSPAN headers are stripped; MPLS labels are left untouched. The header diagram for this case is as follows:

ERSPAN headers				Payload frame headers			
L2	IPv4	GRE	ERSPAN	L2	MPLS	IPv4	TCP

Configuring ERSPAN-II header stripping

Use this task to enable or disable Encapsulated Remote Switch Port Analyzer (ERSPAN-II) header stripping on an Ethernet interface.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command to access Ethernet interface configuration mode.

```
device(config)# interface ethernet 0/2
```

3. Enable or disable this feature on the interface.
   - To enable ERSPAN-II header stripping, enter **strip-erspan**.

```
device(conf-if-eth-0/2)# strip-erspan
```

   - To disable ERSPAN-II header stripping, enter **no strip-erspan**.

```
device(conf-if-eth-0/2)# no strip-erspan
```

The following example enables ERSPAN-II header stripping on an interface.

```
device# configure terminal
device(config)# interface ethernet 0/2
device(conf-if-eth-0/2)# strip-erspan
```

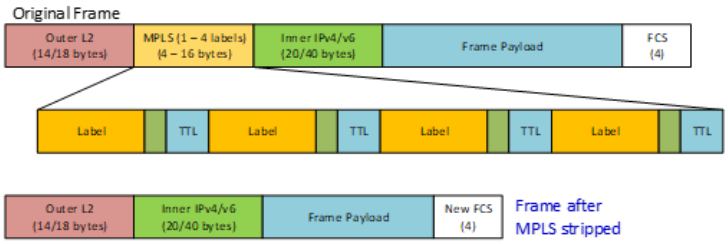
MPLS header stripping

MPLS header stripping is effective for both IP payload and pseudo-wire.

IP payload

The following diagram shows frame structure before and after MPLS header stripping for an encapsulated IP payload. Stripping the MPLS headers prepares the inner headers and frame payload for forwarding and processing.

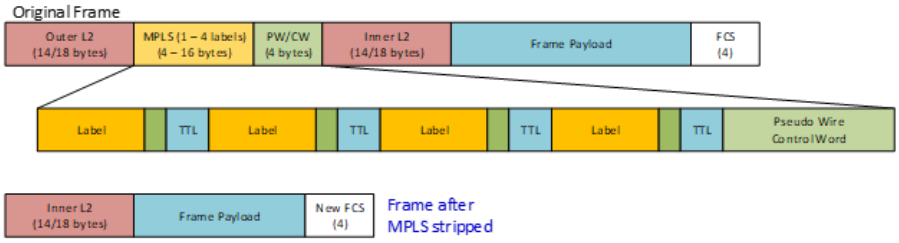
FIGURE 8 Before and after MPLS header stripping (IP payload)



Pseudo-wire

The following diagram shows frame structure before and after MPLS header stripping for an entire encapsulated Ethernet frame (pseudo-wire). Stripping the outer headers (MPLS labels, outer L2, and the pseudo-wire control word) prepares the inner headers and frame payload for forwarding and processing.

FIGURE 9 Before and after MPLS header stripping (pseudo-wire)



Configuration guidelines for MPLS header stripping

For tunneled frames, header-stripping affects only the outer, encapsulation headers. For example, if both VXLAN and MPLS header-stripping are configured, only VXLAN headers are stripped. The header diagram for this case is as follows:

VXLAN headers				Payload frame headers			
L2	IPv4	UDP	VXLAN	L2	MPLS	IPv4	TCP

If both MPLS and 802.1BR or VN-Tag header-stripping are configured, MPLS is always implemented. However, details vary between IP-over-MPLS and L2-over-MPLS regarding 802.1BR or VN-Tags:

- Under IP-over-MPLS, only the MPLS labels are stripped; 802.1BR or VN-Tags are not affected. The header diagram for this case is as follows:

IP-over-MPLS frame headers			
L2 (802.1BR or VN-Tag)	MPLS	IPv4	TCP

- Under L2-over-MPLS (pseudo-wire), both the outer L2 (with 802.1BR or VN-Tag) and the MPLS header are stripped. The header diagram for this case is as follows:

IP-over-MPLS frame (pseudo-wire) headers				
L2 (802.1BR or VN-Tag)	MPLS	L2	IPv4	TCP

Configuring MPLS header stripping

Use this task to enable or disable Multi-Protocol Label Switching (MPLS) header stripping on an Ethernet interface.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command to access Ethernet interface configuration mode.

```
device(config)# interface ethernet 0/2
```

3. Enable or disable this feature on the interface.
   - To enable MPLS header stripping, enter **strip-mpls**.

```
device(conf-if-eth-0/2)# strip-mpls
```

   - To disable MPLS header stripping, enter **no strip-mpls**.

```
device(conf-if-eth-0/2)# no strip-mpls
```

The following example enables MPLS header stripping on an interface.

```
device# configure terminal
device(config)# interface ethernet 0/2
device(conf-if-eth-0/2)# strip-mpls
```

GTP de-encapsulation

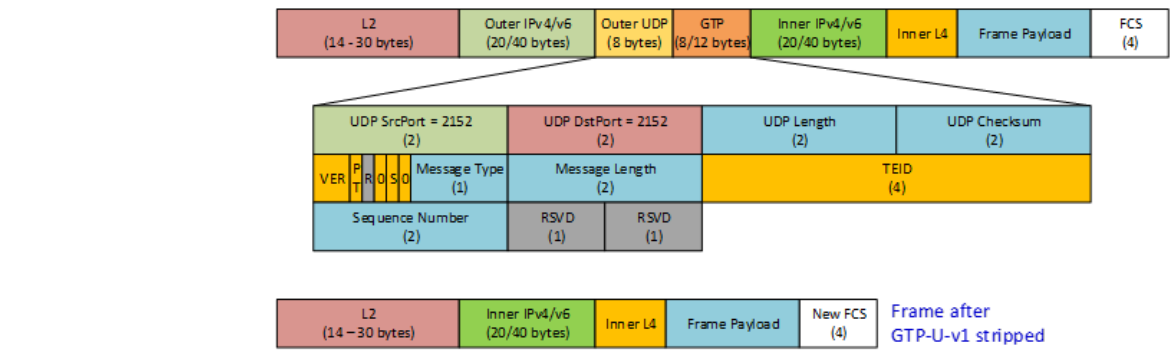
The following figure shows frame structure before and after GTP de-encapsulation.

GTP-U-v1-encapsulated frames have an inner structure of IPv4/v6 tunneled in L2-IPv4/v6-UDP-GTP headers. When GTP de-encapsulation (header-stripping) is enabled, the outer IP, UDP, and GTP headers are discarded, but the outer L2 header is retained.

Following such de-encapsulation, ACLs are applied as follows:

- Layer 2 ACLs apply to the outer L2 header.
- Layer 3 ACLs apply to the inner IP header.

FIGURE 10 Before and after GTP de-encapsulation



GTP de-encapsulation configuration guidelines

GTP de-encapsulation is supported only for GTP v.1 frames, with or without a sequence number.

When GTP de-encapsulation is performed on a frame, only one C-tag is retained in the L2 header. Other tags—802.1BR, VN-Tag, S-Tag, and Outer C-Tag—are dropped from the L2 header.

If GTP de-encapsulation is applied to a frame, VLAN-header modification settings are ignored on that frame. For details, refer to VLAN-header flow modification on page 49.

Configuring GTP de-encapsulation

Use this task to enable or disable GTP de-encapsulation on an Ethernet interface.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command to access Ethernet interface configuration mode.

```
device(config)# interface ethernet 0/2
```

3. Enable or disable this feature on the interface.
   - To enable GTP de-encapsulation, enter **gtp-de-encapsulation**.

```
device(conf-if-eth-0/2)# gtp-de-encapsulation
```

   - To disable GTP de-encapsulation, enter **no strip-vxlan**.

```
device(conf-if-eth-0/2)# no gtp-de-encapsulation
```

The following example enables GTP de-encapsulation on an interface.

```
device# configure terminal
device(config)# interface ethernet 0/2
device(conf-if-eth-0/2)# gtp-de-encapsulation
```

Configuring GTP-HTTPS frame filtering

Use this task to enable and disable dropping GPRS Tunneling Protocol (GTP)-v.1 frames that encapsulate HTTPS packets.

NOTE
For dropped GTP-HTTPS frames, Interface-level header stripping on page 38 is not relevant.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Access a physical interface on which you need to configure this feature.

```
device(conf)# interface ethernet 0/5
```

3. Enable or disable this feature on the interface.
   - To enable dropping GTP frames that encapsulate HTTPS packets, enter **deny inner-gtp-https**.

     ```
     device(conf-if-eth-0/5)# deny inner-gtp-https
     ```

   - To disable dropping GTP frames that encapsulate HTTPS packets, enter **no deny inner-gtp-https**.

     ```
     device(conf-if-eth-0/5)#  no deny inner-gtp-https
     ```

VLAN-header flow modification

NPB supports 802.1q VLAN outer-header stripping and addition for aggregation, replication, and load balancing, as described in the following table. For implementation details, refer to the linked tasks.

VLAN-header modification is implemented at flow level. The other types of header modification under NPB are implemented at interface level:

- Interface-level header stripping on page 38
- Configuring GTP-HTTPS frame filtering on page 48

In general, interface-level header-stripping and VLAN-header modification can apply concurrently. The only exception is GTP de-encapsulation: If GTP de-encapsulation is applied to a frame, VLAN-header modification settings are ignored on that frame.

TABLE 29 VLAN-header modification commands and tasks

Feature	**set** command	Task
Aggregation	**set interface**	Aggregating traffic from interfaces on page 25
Replication	**set next-hop-tvf-domain**	Replicating traffic to multiple interfaces on page 28
Load balancing	**set interface**	Forwarding traffic to a port-channel on page 34

Internal Loopback

Basics of internal loopback

Internal loopback is a software feature supported under NPB to replicate traffic for service-port chaining and other implementations.

Ports configured with internal loopback function as service ports.

Internal-loopback configuration guidelines

Follow these guidelines when implementing internal loopback on a device in NPB mode.

- You can implement internal loopback at physical-interface level.
- You can implement internal loopback at port-channel level.
- There is no upper limit to the number of device ports configured with internal loopback.
- Oversubscription detection is not supported.
- If there is egress from an internal-loopback port, it is ignored.
- Ports supported for internal loopback include:
  - Breakout ports
  - Ports with no SFP or media connection
- Under internal loopback, the following commands are blocked:
  - **fec**
  - **link-error-disable**
  - **link-fault-signaling**
  - **link-up-without-rx**

Port-channel guidelines

The following additional guidelines apply to internal loopback configured on port-channels.

- For interfaces included in a port-channel, implement internal loopback only at port-channel level.
- To avoid speed-mismatch errors, assign a uniform speed to all interfaces in the port-channel, using the **speed** command.
- After attaching a port to a port-channel already configured with loopback phy, the newly attached port is down. To bring it up, run shut and then no shut.
- After removing a port from a loopback phy port-channel, the newly removed port is down. To bring it up, run shut and then no shut.

Configuring internal loopback on a physical interface

Use this task to configure internal loopback on a physical interface.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command to access Ethernet interface configuration mode.

```
device(config)# interface ethernet 0/2
```

3. Enter the **shutdown** command on the interface.

```
device(conf-if-eth-0/2)# shutdown
```

4. Enter the **loopback phy** command to implement internal loopback on that interface.

```
device(conf-if-eth-0/2)# loopback phy
```

5. Enter the **no shutdown** command on the interface.

```
device(conf-if-eth-0/2)# no shutdown
```

Configuring internal loopback on a port-channel

Use this task to configure internal loopback on a port-channel.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface port-channel** command to access port-channel configuration mode.

```
device(config)# interface port-channel 20
```

3. Enter the **shutdown** command on the port-channel.

```
device(config-Port-channel-20)# shutdown
```

4. Enter the **loopback phy** command to implement internal loopback on the port-channel.

```
device(config-Port-channel-20)# loopback phy
```

5. Enter the **no shutdown** command on the port-channel.

```
device(config-Port-channel-20)# no shutdown
```

Internal loopback use cases

There are several NPB internal loopback use cases.

Forward VXLAN traffic

This use case is as follows:

1. Forward traffic with VXLAN headers to a service port (a port on which internal loopback is enabled).
2. Strip the VXLAN headers and forward the traffic to an egress port or port-channel.

Here is a sample implementation, with ### comments preceding implementation stages:

```
### Define a UDA profile
device# configure terminal
device(config)# uda-key profile prof_01
device(conf-uda-profile-prof_01)# flow header0 ETHERNET header1 IPV4 header2 UDP header3 VXLAN
device(conf-uda-profile-prof_01)# uda-key0 header3 VNI
device(conf-uda-profile-prof_01)# end

### Define a UDA for the ingress > service-port route-map.
device(config)# uda access-list extended uda_01
device(conf-uda-acl-ext)# seq 10 permit 0x123 0xfff 0x0 0x0 0x0 0x0 0x0 0x0 count
device(conf-uda-acl-ext)# end

### Define a route-map that will forward VXLAN frames to the service port.
device(config)# route-map to_service_01 permit 10
device(config-route-map-to_service_01/permit/1)# match uda address acl uda_01
device(config-route-map-to_service_01/permit/1)# precedence 10 set interface ethernet 0/10
device(config-route-map-to_service_01/permit/1)# end

### Apply the UDA profile and the route-map to the ingress interface.
device(config)# interface ethernet 0/31
device(conf-if-eth-0/31)# uda-profile-apply prof_01
device(conf-if-eth-0/31)# npb policy route-map to_service_01
device(conf-if-eth-0/31)# no shut
device(conf-if-eth-0/31)# end

### Define an ACL for the service-port > egress-port route-map.
device(config)# mac access-list extended vxlan_mac_01
device(conf-macl-ext)# seq 10 permit any any count
device(conf-macl-ext)# end

### Define a service-port > egress-port route-map
device(config)# route-map to_egress_01 permit 10
device(config-route-map-to_egress_01/permit/10) match mac address acl vxlan_mac_01
device(config-route-map-to_egress_01/permit/10) precedence 10 set interface ethernet 0/32
device(config-route-map-to_egress_01/permit/10) end
device# conf

### Configure the service (internal loopback) port.
device(config)# interface ethernet 0/10
device(conf-if-eth-0/10)# loopback phy
device(conf-if-eth-0/10)# description vxlan_service_port
device(conf-if-eth-0/10)# strip-vxlan
device(conf-if-eth-0/10)# npb policy route-map to_egress_01
device(conf-if-eth-0/10)# no shut
device(conf-if-eth-0/10)# end
device(config)#
```

Onboard Packet Capture

For debugging, Onboard Packet Capture saves frames that ingress or egress a port.

In default system-mode, you can implement most Onboard Packet Capture features, using the **capture packet interface ethernet** command to capture frames ingressing or egressing a physical interface. You can use the **no capture packet interface ethernet** command to stop packet capture.

However, in Network Packet Broker (NPB) system-mode, you can also specify the number of frames you want to capture. After capturing the specified number of frames, packet capture automatically stops.

When packet capture is active, the frames are saved to `/tmp/pktcapture_running.pcap`. Upon termination of packet capture (by reaching **packet-count** or by running **no capture packet interface ethernet**, this file is renamed `/tmp/pktcapture.pcap`. By using the Unix **scp** command, you can securely copy`/tmp/pktcapture.pcap` for analysis.

To display captured frames, use the **show capture packet interface ethernet** command. To display the current capture configuration, use the **show capture packet config** command.

Configuration guidelines

The following configuration guidelines apply to Onboard Packet Capture:

- In default system-mode, control frames and ACL-logged frames are captured.
- In Network Packet Broker (NPB) system-mode, data frames are also captured.
- You run one packet-capture configuration at a time, on a specific physical interface.
- For that interface, you specify ingress or egress traffic.
- You can specify Layer 2 frames, Layer 3 frames, or all frames for which capture is supported for the current system-mode.

NPB Onboard Packet Capture example

The following example captures up to 5000 frames ingressing to an Ethernet interface.

```
device# capture packet interface ethernet 0/9 direction rx filter all packet-count 5000
```

NPB show commands

There are several show commands that display Network Packet Broker (NPB) information, as listed in the following table.

TABLE 30 NPB show commands in the *Command Reference*

Command	Description
show capture packet config	Displays the current packet-capture configuration.
show capture packet interface ethernet	Displays information about captured frames.
show hardware profile current	Displays details of the current active hardware profile, including system-mode and IPv6-address mode.
show inner-gtp-https	Displays a list of all interfaces on which dropping of GPRS Tunneling Protocol (GTP) frames that encapsulate HTTPs packets is enabled.
show interface ethernet	Displays the detailed configuration and capabilities of a specific interface, including internal-loopback status.
show ip interface brief	Indicates which physical and port-channel interfaces are configured for internal loopback.
show packet-encap-processing	Displays information about the interfaces on which header processing is enabled.
show route-map	Displays configuration details of all route maps, of a specific route map, or of the route map applied to an interface.
show running-config tvf-domain	Displays configuration details of one or all TVF domains.