Extreme SLX-OS Network Packet Broker Configuration Guide, 18s.1.03a

Supporting the ExtremeSwitching SLX 9140 and SLX 9240 Switches

9036170-01 Rev AA
October 2020

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, see: www.extremenetworks.com/company/legal/trademarks

Open Source Declarations

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: https://www.extremenetworks.com/support/policies/open-source-declaration/

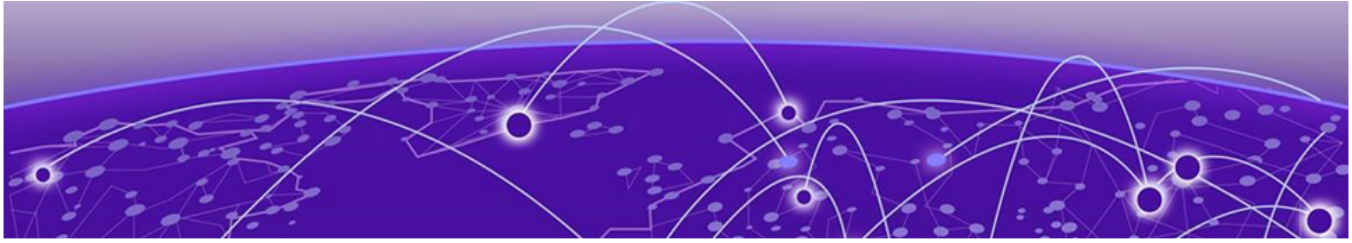


Table of Contents

Preface

This section describes the text conventions used in this document, where you can find additional information, and how you can provide feedback to us.

Text Conventions

Unless otherwise noted, information in this document applies to all supported environments for the products in question. Exceptions, like command keywords associated with a specific software version, are identified in the text.

When a feature, function, or operation pertains to a specific hardware product, the product name is used. When features, functions, and operations are the same across an entire product family, such as ExtremeSwitching switches or SLX routers, the product is referred to as *the switch* or *the router*.

Table 1: Notes and warnings

Icon	Notice type	Alerts you to...
	Tip	Helpful tips and notices for using the product.
	Note	Useful information or instructions.
	Important	Important features or instructions.

Table 1: Notes and warnings (continued)

Icon	Notice type	Alerts you to...
	Caution	Risk of personal injury, system damage, or loss of data.
	Warning	Risk of severe personal injury.

Table 2: Text

Convention	Description
`screen displays`	This typeface indicates command syntax, or represents information as it appears on the screen.
The words *enter* and *type*	When you see the word *enter* in this guide, you must type something, and then press the Return or Enter key. Do not press the Return or Enter key when an instruction simply says *type*.
Key names	Key names are written in boldface, for example **Ctrl** or **Esc**. If you must press two or more keys simultaneously, the key names are linked with a plus sign (+). Example: Press **Ctrl**+**Alt**+**Del**
Words in italicized type	Italics emphasize a point or denote new terms at the place where they are defined in the text. Italics are also used when referring to publication titles.
NEW!	New information. In a PDF, this is searchable text.

Table 3: Command syntax

Convention	Description
bold text	Bold text indicates command names, keywords, and command options.
italic text	Italic text indicates variable content.
[ ]	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, such as passwords, are enclosed in angle brackets.
...	Repeat the previous element, for example, *member*[*member*...].
\	In command examples, the backslash indicates a “soft” line break. When a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash.

Documentation and Training

Find Extreme Networks product information at the following locations:

Current Product Documentation

Release Notes

Hardware and software compatibility for Extreme Networks products

Extreme Optics Compatibility

Other resources such as white papers, data sheets, and case studies

Extreme Networks offers product training courses, both online and in person, as well as specialized certifications. For details, 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: (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, or serial numbers for all involved Extreme Networks products
- A description of the failure
- A description of any actions 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

Subscribe 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 (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. Select **Submit**.

Providing Feedback

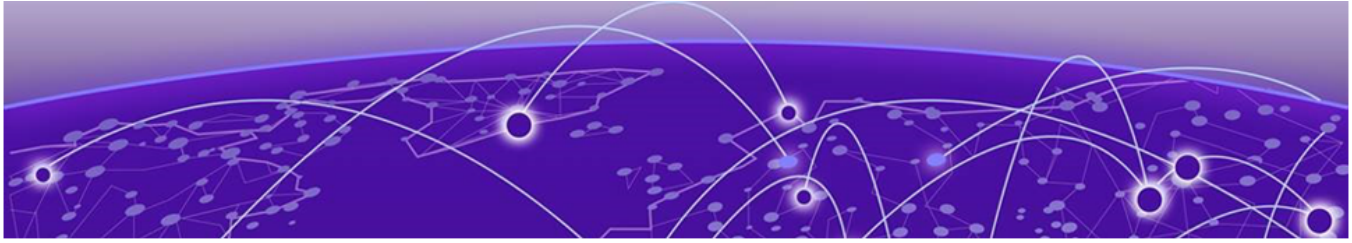
The Information Development team at Extreme Networks has made every effort to ensure the accuracy and completeness of this document. We are always striving to improve our documentation and help you work better, so we want to hear from you. We welcome all feedback, but we especially want to know about:

- Content errors, or confusing or conflicting information.
- Improvements that would help you find relevant information in the document.
- Broken links or usability issues.

If you would like to provide feedback, you can do so in three ways:

- In a web browser, select the feedback icon and complete the online feedback form.
- Access the feedback form at https://www.extremenetworks.com/documentation-feedback/.
- Email us at documentation@extremenetworks.com.

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.



ACLs and UDAs under NPB

Stanza and ACL 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.

In NPB mode, ACLs within route-maps 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* :

Route-maps contain **`match { ip | ipv4 } address acl`** statements. (NPB route-maps can also contain **`match uda`** 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 route-maps, permit and deny rules specify criteria for route-map decisions.
- 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.

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 { ip | ipv6 } address acl** or **match uda** 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

- Layer 2: **match mac address acl**
- Layer 3: **match { ip | ipv6 } address acl**
- User-defined ACLs: **match uda acl**
- User-defined ACLs: **match uda**

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

About This Task

Procedure

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

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

4. 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

About This Task

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.

Procedure

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

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

- 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.

Note
On NPB-grid devices, only the **set interface pbf-destination-group** option is supported.

- 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

Table 5: ETHERNET (Ethernet header)

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

Table 5: ETHERNET (Ethernet header) (continued)

Field	Width (Bits)	Description
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
	32	Source and destination port

Table 11: UDP (UDP header) (continued)

Field	Width (Bits)	Description
PORTS		
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)

Table 16: MPLS (MPLS header) (continued)

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

Table 20: PAYLOAD4 (4 bytes) (continued)

Field	Width (Bits)	Description
WORD0		

Table 21: PAYLOAD8 (8 bytes)

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

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

Procedure

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
```

Example

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

Procedure

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
device(config)# interface ethernet 1/2
```

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
device(conf-if-eth-1/2)# uda-profile-apply uda_gtp
```

Applying a UDA profile on a port-channel interface

Procedure

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

Creating a UDA

Procedure

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

3. Enter rules, specifying the needed parameters.

```
device(conf-uda-acl-ext)# permit 0x0800 0xFFFF 17 0xFF 3400 0xFFFF 0x11223344
0xFFFFFFFF
device(conf-udaacl-ext)# permit 0x08680000 0xffff0000 0x11000000 0xff000000 0x13c40000
        0xffff0000 0x4d455353 0xffffffff count
```

Example

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 count
device# configure terminal
device(config)# uda access-list extended uda_01
device(conf-udaacl-ext)# permit 0x00001111 0x0000ffff 0x00002222 0x0000ffff 0x00003333
0x0000ffff 0x00004444 0x0000ffff count
```

UDA examples

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
```

UDAs

User-defined ACLs (UDAs) examine packet fields at specified offsets, applying permit and deny rules.

Basics of user-defined ACLs (UDAs)

UDAs are supported for NPB:

- Applied to physical and port-channel interfaces with the **uda access-group** command, as described in this section.
- For policy-based routing (PBR), as described in "Policy-Based Routing under NPB."

UDA configuration guidelines

The following general guidelines apply to UDA configuration:

- UDAs are supported only if the npb-optimised-1 profile is enabled.
- You can apply UDAs only on physical and port-channel interfaces, but not on virtual interfaces.
- UDAs are not supported as receive ACLs (rACLS).
- The processing priority of UDAs is lower than other ACLs: OpenFlow > rACLs > PBR > Layer 3 ACLs > Layer 2 ACLs > UDAs.
- You cannot apply more than one UDA profile to a group of interfaces that share an ASIC. For details, refer to the "Association of ports to ASICs" tables in the *SLX 9850 Router Technical Specifications*.
- TCAM sharing is not supported for UDAs.

UDA-profile guidelines

For the **uda-key profile** > **uda-offsets** > **packet-start** option, the offset begins with the Ethernet header.

The following **uda-key profile** guidelines apply to **uda-offsets**> **first-header**, **second-header**, **third-header**, or **fourth-header**:

- The Ethernet header (including VLAN header) is not considered.
- The profile works equally for tagged and untagged packets. For example, for both of the following packets, **uda-offsets** considers <IPv6> as the first header and <UDP> as the second header:
  - (Untagged) <MAC><IPV6><UDP>
  - (Tagged) <MAC><VLAN><IPV6><UDP>
- To match fields inside a tunnel header (for example, GTP, VXLAN, or GRE), define that header as the offset base.

UDA hardware support and scaling

UDAs are supported on all interface-module line cards available for the SLX 9850 device.

For UDA and UDA-rule scaling limits, refer to the "ACL and rule limits" topic in the *Extreme SLX-OS Security Configuration Guide*.

UDA-profile design

This analysis is especially important for tunnel-protocol elements like GTP, VXLAN, and GRE, which are not parsed. Before you begin, examine the following *Extreme SLX-OS Command Reference* topics:

- **uda-key profile**

- **uda-offsets**

Table 24: UDA-profile example

Outer header	Inner header	Field	Field-value	Base and offset
Ethernet VLAN				
IPv6 (1st)				
UDP (2nd)		Outer UDP dest. port	0x0868	second-header 2
GTP (3rd)	IPv6	Inner IPv6	0x11	third-header 14
	UDP	Inner UDP dest. port	0x13c4	third-header 50
		Data	0x4D455353	third-header 56

The following topics implement this design on a physical interface:

- Configuring a UDA profile on page 23
- Applying a UDA profile on a physical interface on page 19
- Creating a UDA on page 20
- Applying a UDA on a physical interface on page 25

Configuring a UDA profile

Before You Begin

Procedure

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 uda_gtp
```

3. Enter the **uda-offsets** command to configure the profile.

```
device(config-uda-key)# uda-offsets second-header 2 third-header 14 third-header 50
third-header 56
```

Example

The following example uses the second header (UDP) as the offset base and specifies 49 as the offset.

```
device# configure terminal
device(config)# uda-key profile uda-prof_02
device(config-uda-key)# uda-offsets second-header 49 ignore ignore ignore
```

Applying a UDA profile on a physical interface

Procedure

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
device(config)# interface ethernet 1/2
```

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
device(conf-if-eth-1/2)# uda-profile-apply uda_gtp
```

Applying a UDA profile on a port-channel interface

Procedure

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

Creating a UDA

Procedure

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

3. Enter rules, specifying the needed parameters.

```
device(conf-uda-acl-ext)# permit 0x0800 0xFFFF 17 0xFF 3400 0xFFFF 0x11223344
0xFFFFFFFF
device(conf-udaacl-ext)# permit 0x08680000 0xffff0000 0x11000000 0xff000000 0x13c40000
        0xffff0000 0x4d455353 0xffffffff count
```

Example

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 count
```

```
device# configure terminal
device(config)# uda access-list extended uda_01
device(conf-udaacl-ext)# permit 0x00001111 0x0000ffff 0x00002222 0x0000ffff 0x00003333
0x0000ffff 0x00004444 0x0000ffff count
```

Applying a UDA on a physical interface

Procedure

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 5/2
```

3. Enter the **uda access-group** command—specifying the UDA and ingress direction—to apply the UDA on the interface.

```
device(conf-if-eth-5/2)# uda access-group uda_gtp in
```

Applying a UDA on a port-channel interface

Procedure

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 access-group** command, specifying the specifying the UDA and the ingress direction.

```
device(config-Port-channel-10)# uda access-group uda_02 in
```

UDA-on-interface example

Note
For UDA matching in route-maps, refer to "Configuring route-maps with UDAs."

The goal of this use case is to enable ingress traffic with UDP header = ESP (Encapsulating Security Protocol) to be forwarded to monitoring and analytics tools.

Packet details

- Ethernet Type: 0x8100 (802.1Q)
- 802.1Q Type: 0x0800 (IPv4)
- IP Protocol: 17 (UDP)
- UDP: D. Port 2152

UDP payload

- GPRS Message Type: 0xff (T-PDU)
- IP Protocol: 47 (GRE)
- GRE Type: 0x0800 (IP)
- IP Protocol: 0x32 (ESP)

Match: ESP Packets (encapsulating sec. pay)

- IP Protocol Type 50
- 1 byte
- 0x32 (ESP)
- Byte 87 (from start of packet)

Task 1: Configure a UDA profile

Procedure

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 uda-prof_03
```

3. Enter the **uda-offsets** command to configure the profile.

```
device(config-uda-key)# uda-offsets second-header 49 ignore ignore ignore
```

Note
UDP is the second header. The offset value is 49 from the start of the UDP header.

Task 2: Create a UDA ACL

Procedure

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_03
```

3. Enter a permit rule to validate 0x32 (Encapsulating Security Protocol [ESP]) at the 49th position from the UDP header.

```
permit 0x32 0xff 0x0 0x0 0x0 0x0 0x0 0x0
```

Task 3: Apply the UDA profile and the UDA on an interface

Procedure

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 1/2
```

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

```
device(conf-if-eth-1/2)# uda-profile-apply uda-prof_03
```

4. Enter the **uda access-group** command—specifying the UDA and ingress direction—to apply the UDA on the interface.

```
device(conf-if-eth-5/2)# uda access-group uda_03 in
```

UDA show and clear commands

Table 25: UDA show commands in the *Command Reference*

Command	Description
show access-list	For an ACL type and inbound/outbound direction, displays ACL information. You can show information for a specified ACL or only for that ACL on a specific interface. You can also display information for all ACLs bound to a specific interface.
show running-config uda access-list	Displays a list of user-defined ACLs (UDAs) defined on the device—or a specific UDA—including the rules they contain.
show running-config uda-key profile	Displays a list of user-defined ACL (UDA)-profiles defined on the device—or a specific UDA profile.
show statistics access-list	For an ACL type and inbound/outbound direction, displays statistical information—for ACL rules that include the **count** keyword. You can show statistics for a specific ACL or only for that ACL on a specific interface. You can also display statistical information for all ACLs bound to a specific interface.

Table 26: UDA clear commands in the *Command Reference*

Command	Description
clear counters access-list	For an ACL type and inbound/outbound direction, clears ACL statistical information. You can clear all statistics for a specific ACL or only for that ACL on a specific interface. You can also clear statistical information for all ACLs bound to a specific interface.

Traffic Aggregation and Replication

Aggregating traffic from interfaces (single-NPB)

About This Task

Note
For the NPB-grid implementation, refer to Aggregating traffic from TAPs on page 77.

Procedure

1. Configure a regular ACL or a user-defined ACL (UDA):
   - For a regular ACL, refer to Creating ACLs for NPB on page 11.
     ```
     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 12.
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
     ```

Example

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 (single-NPB)

Note
For the NPB-grid implementation, refer to Replicating traffic to multiple NPB-grid destinations on page 75.

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

Procedure

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

Procedure

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

Procedure

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 (single-NPB)

About This Task

Note
For the NPB-grid implementation, refer to Replicating traffic to multiple NPB-grid destinations on page 75.

Procedure

1. Create needed TVF domains, as described in Creating TVF domains on page 30.

```
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 30.)
     ```
     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 11.
     ```
     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 12.
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
     ```

Example

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 (Single-NPB)

Load balancing overview

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.

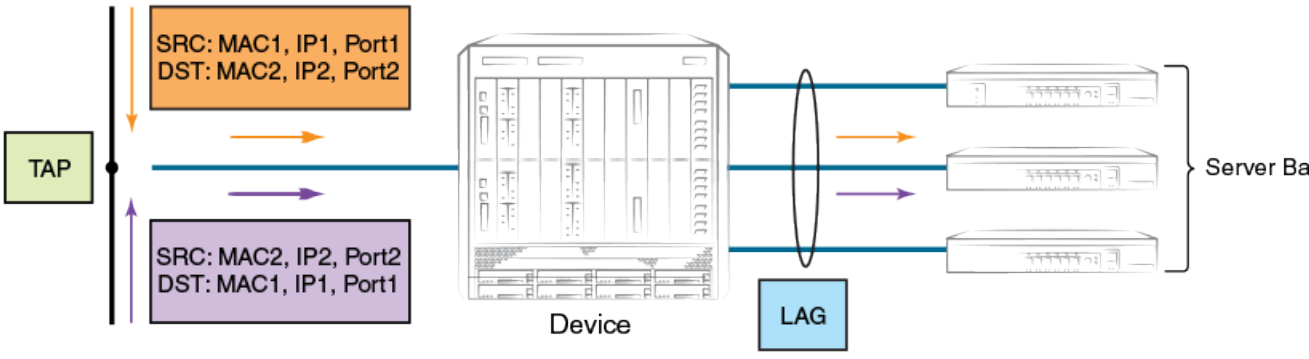


Figure 1: Symmetric load balancing

Load balancing of tunneled frames, with header stripping

- 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 40.

Configuring symmetric load balancing

Procedure

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

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 27: 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 28: 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
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 29: 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 30: 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 31: 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

About This Task

Note
For the NPB-grid implementation, refer to Forwarding NPB-grid traffic on page 74.

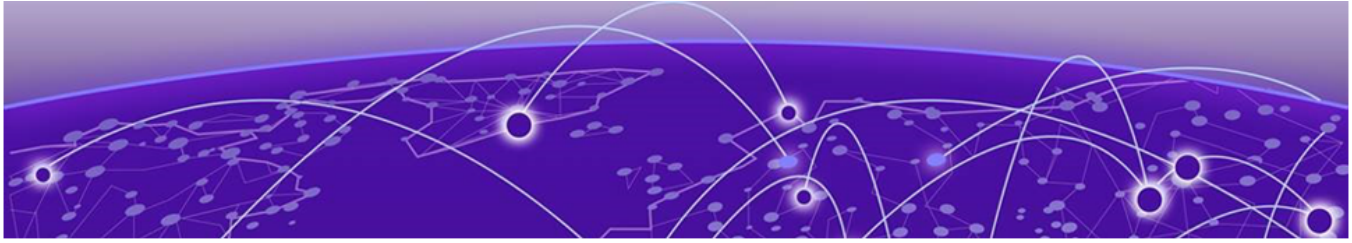
Procedure

1. Configure a regular ACL or a user-defined ACL (UDA):
   - For a regular ACL, refer to Creating ACLs for NPB on page 11.
     ```
     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 12.
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
     ```

Example

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

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 40
- Configuring GTP-HTTPS frame filtering on page 50
- VLAN-header flow modification on page 50

Header-modification flow

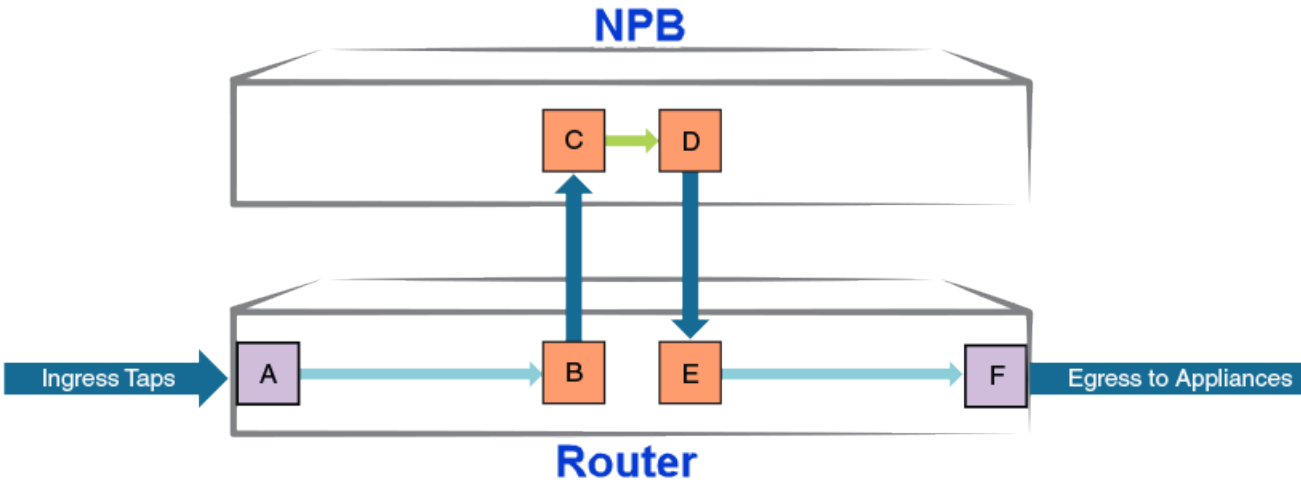


Figure 2: Header-modification flow

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 #unique_54.)
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

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 41
- VN-Tag header stripping on page 42
- VXLAN header stripping on page 43
- NVGRE header stripping on page 45
- ERSPAN-II header-stripping on page 46
- MPLS header stripping on page 47
- GTP de-encapsulation on page 49

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 34.

802.1BR header stripping

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

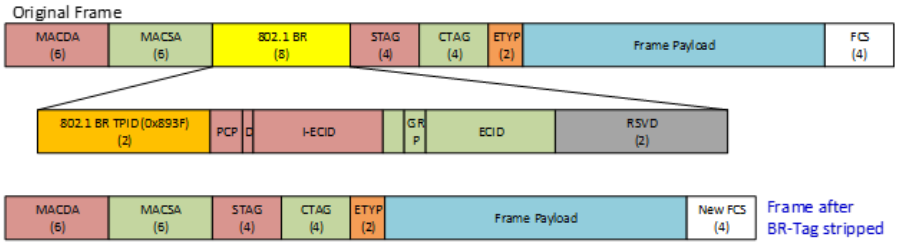


Figure 3: Before and after 802.1BR header stripping

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 42 and Configuring VN-Tag header stripping on page 43.

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

Procedure

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
```

Example

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

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

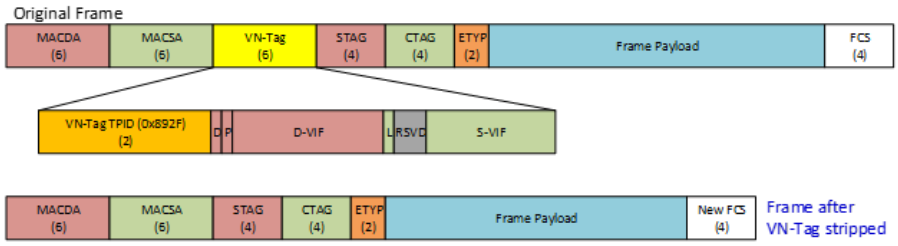


Figure 4: Before and after VN-Tag header stripping

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 42 and Configuring VN-Tag header stripping on page 43.

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

Procedure

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
```

Example

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

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.

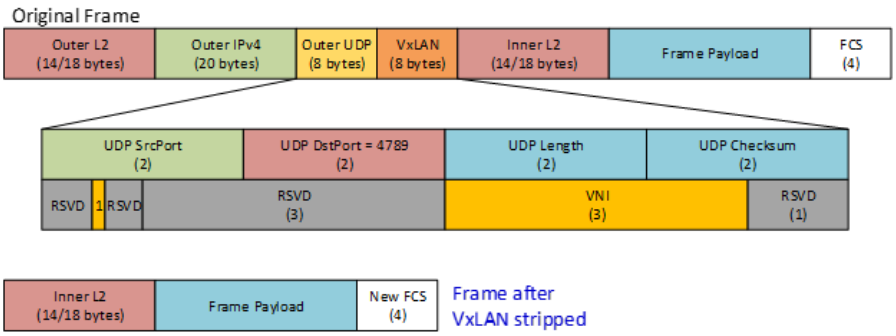


Figure 5: Before and after VXLAN header stripping

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

Procedure

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
```

Example

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

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.

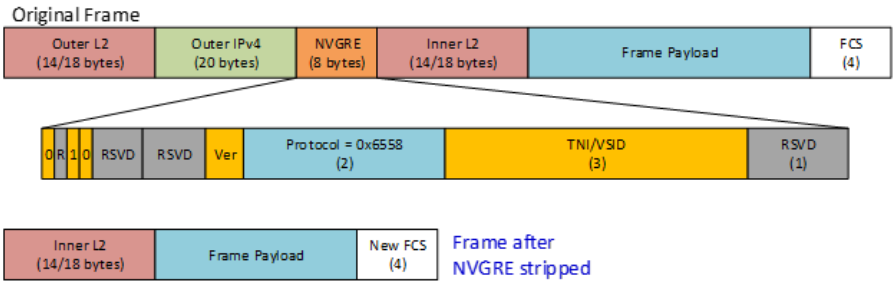


Figure 6: Before and after NVGRE header stripping

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

Procedure

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
```

Example

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

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.

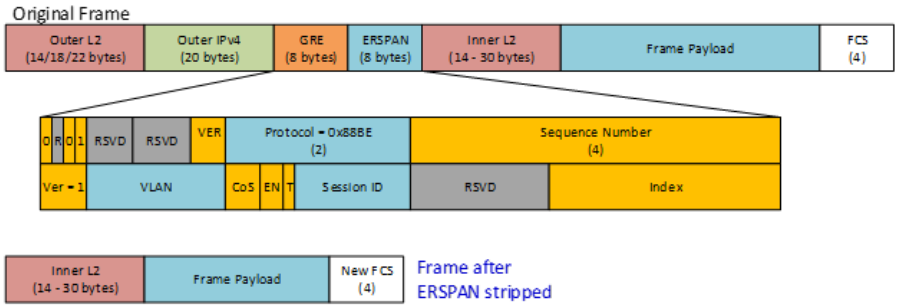


Figure 7: Before and after ERSPAN-II header-stripping

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

Procedure

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
     ```

Example

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

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.

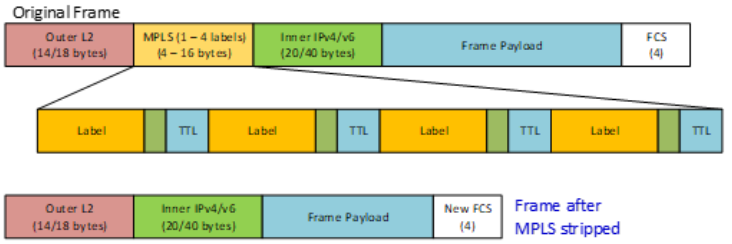


Figure 8: Before and after MPLS header stripping (IP payload)

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.

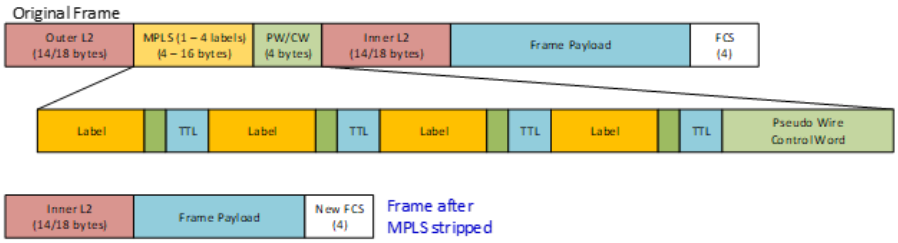


Figure 9: Before and after MPLS header stripping (pseudo-wire)

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

Procedure

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
```

Example

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

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.

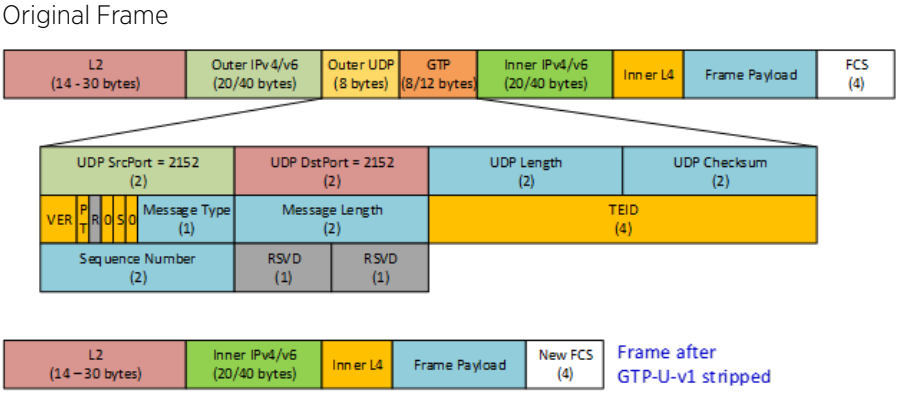


Figure 10: Before and after GTP de-encapsulation

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 50.

Configuring GTP de-encapsulation

Procedure

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
```

Example

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

About This Task

Note
For dropped GTP-HTTPS frames, Interface-level header stripping on page 40 is not relevant.

Procedure

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

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 40
- Configuring GTP-HTTPS frame filtering on page 50

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 32: VLAN-header modification commands and tasks

Feature	Single-NPB tasks	NPB-grid tasks
Forwarding	Forwarding traffic to a port-channel on page 37 (load balancing)	Forwarding NPB-grid traffic on page 74
Aggregation	Aggregating traffic from interfaces (single-NPB) on page 28	Aggregating traffic from TAPs on page 77
Replication	Replicating traffic to multiple interfaces (single-NPB) on page 31	Replicating traffic to multiple NPB-grid destinations on page 75

Internal Loopback

Basics of internal loopback

Ports configured with internal loopback function as service ports.

Note
Internal loopback is not supported for NPB Grids on page 60.

Internal-loopback configuration guidelines

- 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

Procedure

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

Procedure

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

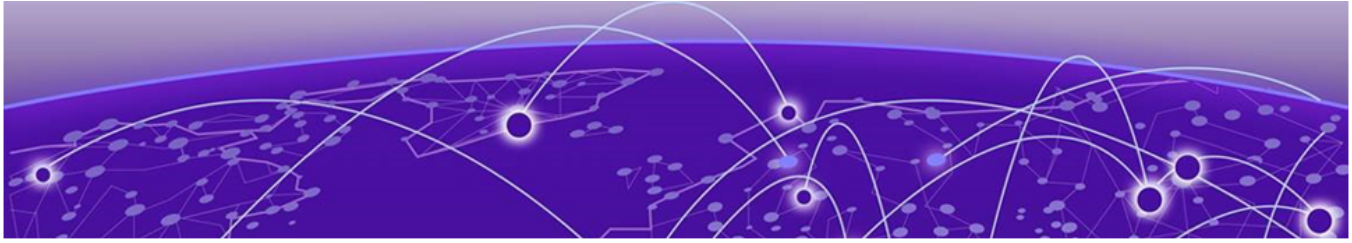
Forward VXLAN traffic

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

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

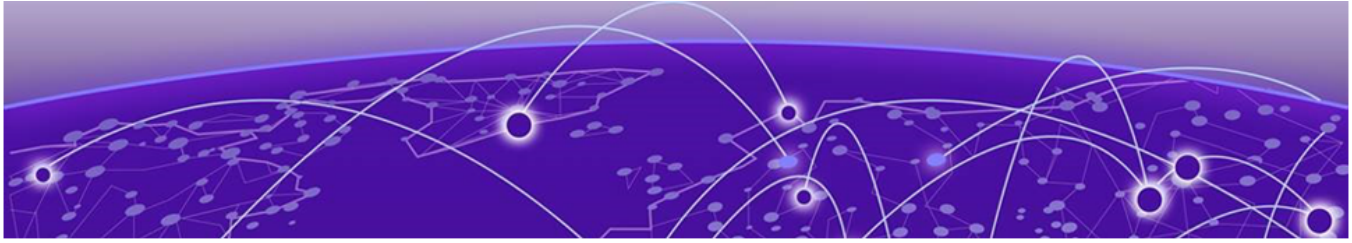
The following configuration guidelines apply to Onboard Packet Capture:

- 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
```



Egress Packet Truncation

Egress packet truncation overview

Truncation is performed at the 'flow' level. The truncation functionality is achieved by defining a truncation profile, which contains a user-specified interface and truncation size, and assigning it to a route-map. The truncation profile determines the final length of the egress frames on the selected flows.

Truncation implementation flow

1. Configure the interface to be used in the truncation profile into loopback mode.
2. Create a truncation profile.
3. Create a route-map.
4. In the route-map, specify the egress interface and the truncation profile.

Truncation guidelines and limitations

The following guidelines and limitations apply to egress packet truncation:

- To reference an interface in a truncation profile, it must first be placed into loopback mode.
- Only four truncation profiles can be created and referenced in route maps.
- A route-map or route-map stanza can reference a single truncation profile any number of times.
- On supported platforms, 10 bytes—including the 4-byte FCS—are appended to the frame after truncation. For example, if you set the truncation size as 256 bytes, the frame size on the wire will be 266 bytes.
- Make sure that in the route map your **`set interface`** or **`set next-hop-tvf-domain`** command correctly specifies a truncation profile that you previously configured. Otherwise, the system automatically provisions the profile that you specified, but with a truncation size of 0. Matching traffic is dropped until you fully configure the profile.

Configuring a truncation profile

Procedure

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

```
device# configure terminal
```

2. Configure internal loopback on a physical interface or port-channel.
   - For configuring internal loopback on a physical interface, refer to Configuring internal loopback on a physical interface on page 53.

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

   - For configuring internal loopback on a port-channel, refer to Configuring internal loopback on a port-channel on page 53.
3. Exit to global configuration mode.

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

4. Enter the **truncation-profile** command to create a truncation profile.

```
device(config)# truncation-profile udp_truncation
```

Note
Names range from 1 through 31 ASCII characters in length, and must start with an alphanumeric character. No special characters are allowed except underscore and hyphen.

5. Enter the **set interface** command to define the physical interface or port-channel for truncation. Only one interface can be specified within a truncation profile. The last entered interface is assigned to the profile.

Note
To reference an interface in a truncation profile, it must first be placed into loopback mode.

```
device(conf-truncation-profile-udp_truncation)# set interface ethernet 0/2
```

6. Enter the **frame-size** command to specify the truncated packet length, in bytes. For example, if you configure the frame-size as 256 bytes, the packets are truncated to 266 bytes on the wire.

Note
The valid range is from 64 through 9216, and must be in multiples of 16 (64, 80, 96...).

```
device(conf-truncation-profile-udp_truncation)# frame-size 256
```

Example

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

```
device# configure terminal
device(config)# truncation-profile udp_truncation
device(conf-truncation-profile-udp_truncation)# set interface ethernet 0/2
device(conf-truncation-profile-udp_truncation)# frame-size 256
```

Referencing a truncation profile in a route map

Procedure

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

```
device# configure terminal
```

2. Enter the **route-map** command to create a route-map.

```
device(config)# route-map filter_udp_tcp permit 10
```

3. In the route-map, enter the **set interface** or **set next-hop-tvf-domain** command, specifying the egress interface and a truncation profile that you previously defined.

```
device(config-route-map-filter_udp_tcp/permit/10)# set interface ethernet 0/3
truncation-profile tcp_truncation
```

Note
On NPB-grid devices, the **set interface pbf-destination-group** option is also supported.

Example

In this example, packets are forwarded to the truncation profile tcp_truncation for truncation and then forwarded to the port-channel 10.

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

- Without stripping 802.1q VLAN tags:

```
device(config-route-map-npb_map1/permit/1))# set interface port-channel 10 truncation-
profile truncation1
```

- Stripping 802.1q VLAN tags:

```
device(config-route-map-npb_map1/permit/1))# set interface port-channel 10 truncation-
profile truncation1 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 port-channel 10 truncation-
profile truncation1 add-vlan outer 2
```

Example

In this example, packets are forwarded to the truncation profile tcp_truncation for truncation and then forwarded to TVF domain 50.

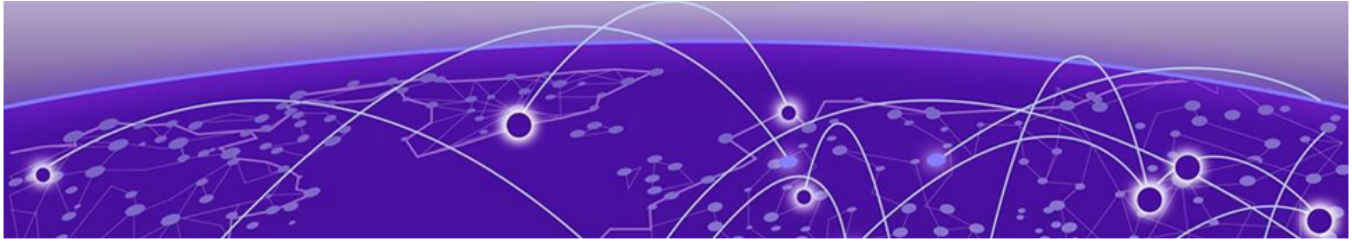
```
device# configure terminal
device(config)# route-map filter_udp_tcp permit 10
device(config-route-map-filter_udp_tcp/permit/10)# match ip address acl tcp_filter
device(config-route-map-filter_udp_tcp/permit/10)# set next-hop-tvf-domain 50 truncation-
profile tcp_truncation
```

Example

In this example, packets are forwarded to the truncation profile udp_truncation for truncation and then forwarded to PBF destination group 1000.

```
device# configure terminal
device(config)# route-map filter_udp_tcp permit 10
```

```
device(config-route-map-filter_udp_tcp/permit/10)# match ip address acl udp_filter
device(config-route-map-filter_udp_tcp/permit/10)# set interface pbf-destination-group
1000 truncation-profile udp_truncation
```



NPB Grids

NPB-grid overview

NPB grids increase the scale of network packet-brokers from one device to multiple aggregation and distribution devices. An NPB grid is a network of:

- Optical-TAP aggregators forwarding data to distributor devices
- Distribution devices forwarding traffic to visibility and monitoring destinations. Ring networks of multi-layer distribution devices are also supported.

Extreme Visibility Manager is the management and provisioning solution for Extreme Network Packet Brokers. Although you can manage a single SLX-OS NPB without Visibility Manager, it is required for NPB grids. Management of NPB grids by Visibility Manager enables:

- Grid configuration and management
- Dynamic best-route determination from TAP inputs to tool destinations
- Load balancing
- NPB-grid statistics

NPB-grid implementation flow

You implement an NPB grid as follows:

1. Prepare charts or tables indicating optimal and backup paths from each aggregator to all relevant tool destinations. This preparation is required for:
   - Configuration of destinations on aggregators and distributors
   - Visibility Manager configuration
2. On each aggregator and distributor, set the global MTU for jumbo frames.
3. On each aggregator and distributor, configure all relevant destinations and destination groups. (For load balancing, you also configure destination loadbalance-groups.) Although you specify identical destination/destination-group IDs on all devices, the **`set interface`** destination command varies, specifying next hops along the primary and backup paths to the tool destination.

4. On each aggregator and distributor, configure identical telemetry profiles and telemetry-collector objects.
5. On each TAP interface, perform the relevant task under TAP configurations on page 74. Each of these tasks includes a **`set interface pbf-destination-group`** command.
6. Make sure that all of the devices in the grid are cabled in accordance with your NPB-grid design.
7. Configure, manage, and monitor the grid from Visibility Manager, as described in the *Extreme Visibility Manager Administration Guide*.

NPB-grid diagrams

Note

For a diagram that includes the Extreme Visibility Manager elements, refer to NPB-grid telemetry for Visibility Manager on page 67.

In the following basic diagram:

- In destination configurations on distributor devices, the **`set interface`** statements specify interfaces leading directly to tool destinations (not to other distributors).
- For graphic simplicity, the destination groups are segregated by distributor. However, you can include destinations from multiple distributors in a single destination group.

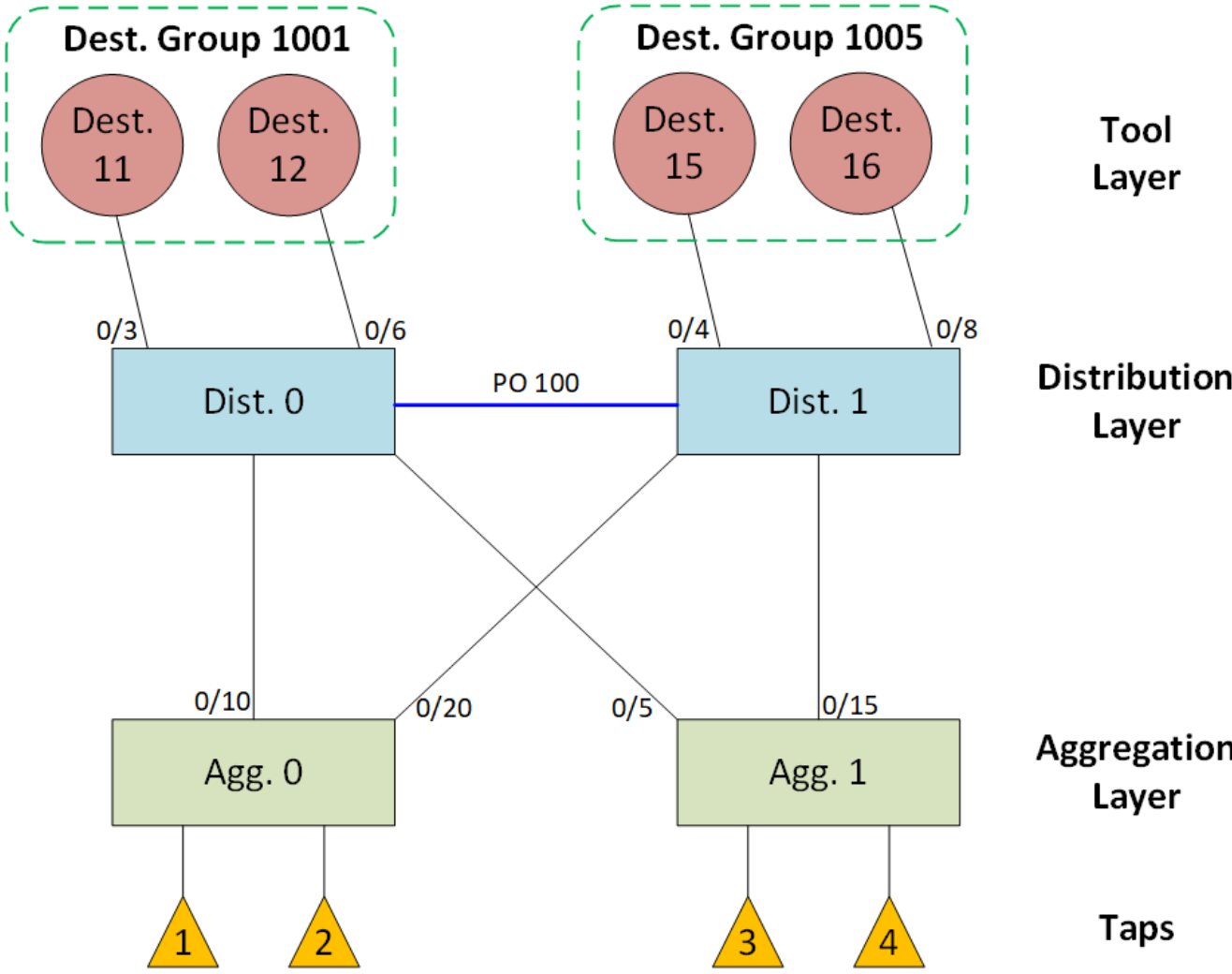


Figure 11: Basic NPB-grid diagram

The following diagram includes paths with and without distributor-to-distributor forwarding.

- The highest priority path for destination 11 would include a **`set interface ethernet 0/3`** statement on Dist. 0.
- The highest priority paths for destinations 15 and 16 would include **`set interface port-channel 12 type-fabric`** statements on Dist. 1.

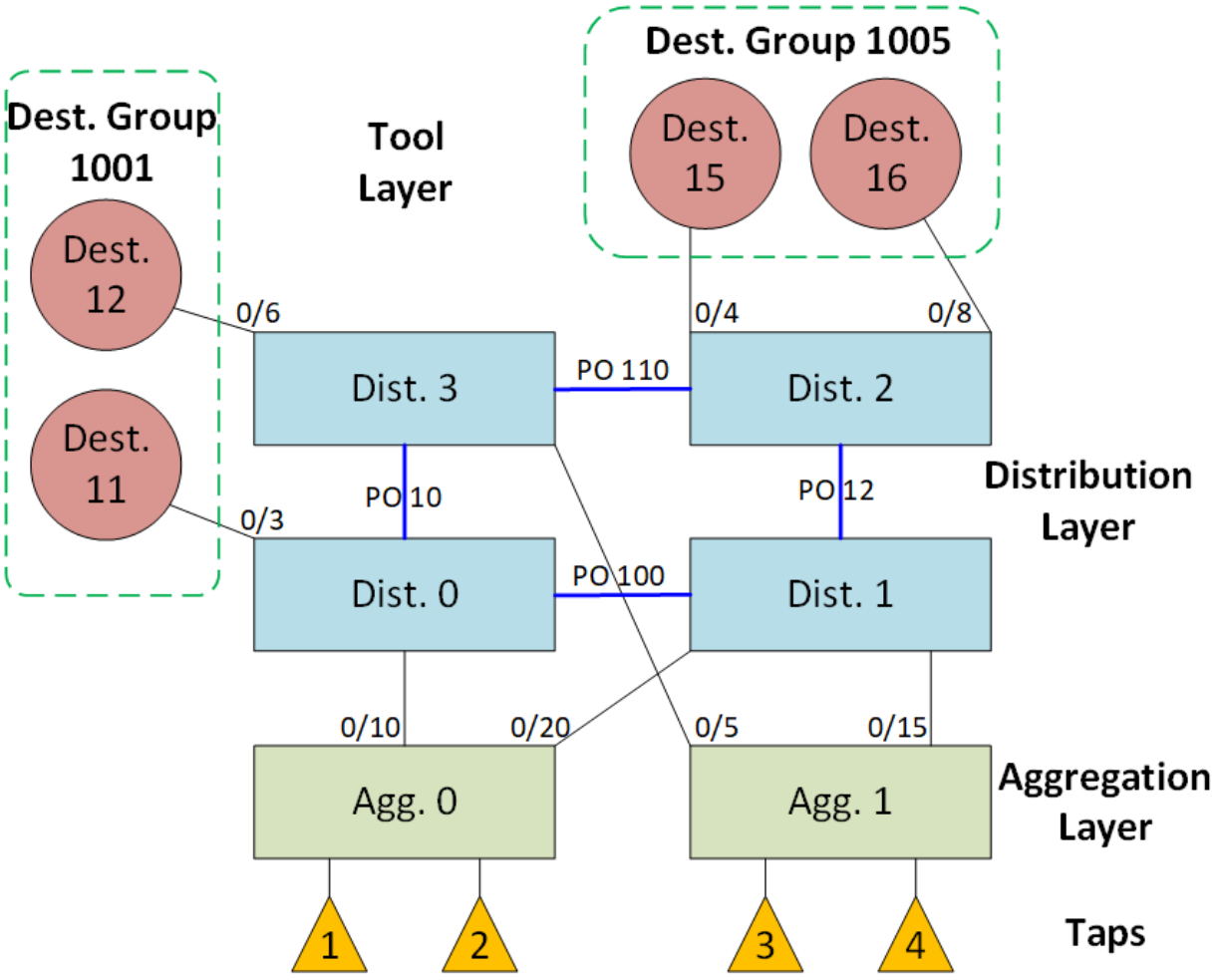


Figure 12: NPB-grid diagram (multi-layer distribution devices)

The following diagram has both types of PBF destination groups:

- Without any load-balancing groups
- With a load-balancing group

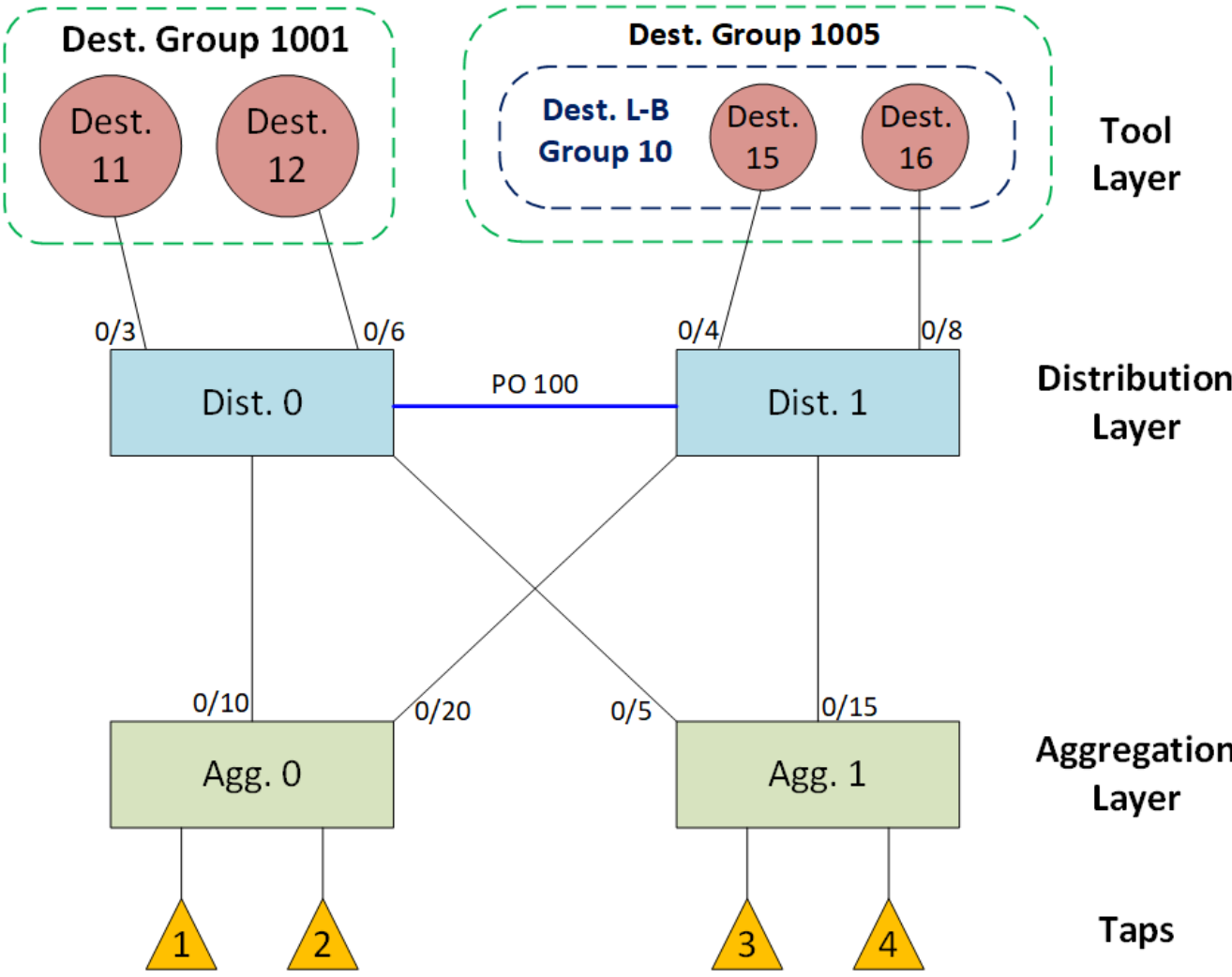


Figure 13: NPB-grid load-balancing diagram

Setting global MTU for jumbo frames

Procedure

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

```
device# configure terminal
```

2. Set the MTU value to 9216.

```
device(config)# mtu 9216
```

Note

Because NPB Grid adds a 38-byte header, maximum frame size is 9178 bytes (9216–38).

Example

The following example sets the global MTU to 9216.

```
device# configure terminal
device(config)# mtu 9216
```

Destination configurations

Note
The commands for creating destinations and destination-groups contain a "pbf" element, which refers to policy-based forwarding. These commands are currently supported only in NPB system-mode.

Destination objects contain one or more **set interface** statements, which specify either:

- A next-hop within the fabric. For this option, include the **type-fabric** keyword in the all of the destination **set interface** statements.
- An edge interface egressing to a monitoring/analysis tool. For this option, do not include the **type-fabric** keyword in any of the destination **set interface** statements.

Configuring PBF destinations

Procedure

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

   ```
   device# configure terminal
   ```

2. (Identical for all devices on any possible path to the tool destination) Enter the **pbf destination** command, specifying the destination ID.

   ```
   device(config)# pbf destination 20
   ```

3. (Identical for all devices on any possible path to the tool destination) To enable statistics for the destination, enter **statistics-enable**.

   ```
   device(config-destination-20)# statistics-enable
   ```

4. Enter one or more **set interface** commands to specify the physical or port-channel interface to the next hop or the actual egress to the destination.
   - For a fabric next-hop, include the **type-fabric** keyword.

     ```
     device(config-destination-20)# precedence 10 set interface ethernet 0/5 type-fabric
     device(config-destination-20)# precedence 30 set interface ethernet 0/6 type-fabric
     ```

     Note
     The **precedence** parameter determines the primary path and precedence among secondary paths. The **set interface** statement with the lowest **precedence** number is the primary path, and so forth.

   - For edge egress to the destination, do not include the **type-fabric** keyword.

     ```
     device(config-destination-20)# precedence 10 set interface ethernet 0/1
     device(config-destination-20)# precedence 30 set interface ethernet 0/3
     ```

Example

The following example creates and configures an edge PBF destination, with statistics enabled.

```
device# configure terminal
device(config)# pbf destination 20
device(config-destination-20)# statistics-enable
device(config-destination-20)# precedence 10 set interface ethernet 0/3
```

Configuring load-balance groups

About This Task

Procedure

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

   ```
   device# configure terminal
   ```

2. Enter the **pbf destination-loadbalance-group** command, specifying the ID.

   ```
   device(config)# pbf destination-loadbalance-group 1000
   ```

3. Add PBF destinations to the load-balance group.

   ```
   device(config-destination-loadbalance-group-1000)# add destination 1-4,20,22
   ```

Example

The following example creates or accesses a load-balance group and adds destinations to it.

```
device# configure terminal
device(config)# pbf destination-loadbalance-group 1000
ddevice(config-destination-loadbalance-group-1000)# add destination 15,25,35
```

Configuring PBF destination-groups

About This Task

Note
Make sure to define identical destination groups on every relevant aggregator and distributor of the NPB grid.

Procedure

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

   ```
   device# configure terminal
   ```

2. Enter the **pbf destination-group** command, specifying a PBF destination-group ID.

   ```
   device(config)# pbf destination-group 100
   ```

3. To add PBF destinations to the destination group, enter the **add destination** command.

   ```
   device(config-destination-group-100)# add destination 10,20,30
   ```

4. To add load-balance groups to the destination group, enter the **add destination-loadbalance-group** command.

   ```
   device(config-destination-group-100)# add destination-loadbalance-group 1000-1003
   ```

5. To remove destinations from a destination group, use the relevant command.
   - To remove all destinations, use the **no add destination** command.

```
device(config-destination-group-100)# no add destination
```

   - To remove some destinations, use the **remove destination** command.

```
device(config-destination-group-100)# remove destination 10,30
device(config-destination-group-100)# do show running-config pbf destination-group
100
  pbf destination-group 100
    add destination 20
!
```

Example

The following example creates or accesses a PBF distribution group and adds destinations to it.

```
device# configure terminal
device(config)# pbf destination-group 100
device(config-destination-group-100)# add destination 1-4,20,22
device(config-destination-group-100)# add destination-loadbalance-group 1000-1003
```

NPB-grid telemetry for Visibility Manager

Note
For the general principles of SLX-OS telemetry, refer to the *Extreme SLX-OS Monitoring Configuration Guide*.

Telemetry *profiles* are basic elements of the various SLX-OS telemetry implementations. Each profile is designed to monitor a specific grouping of data or events, for example, LLDP or PBR.

For transmission of profile-filtered data to Visibility Manager, NPB grids use the SLX-OS external-collector telemetry implementation. In the identical **telemetry collector** objects defined on each aggregation and distribution device, you specify the IP/port of the Visibility Manager telemetry collector.

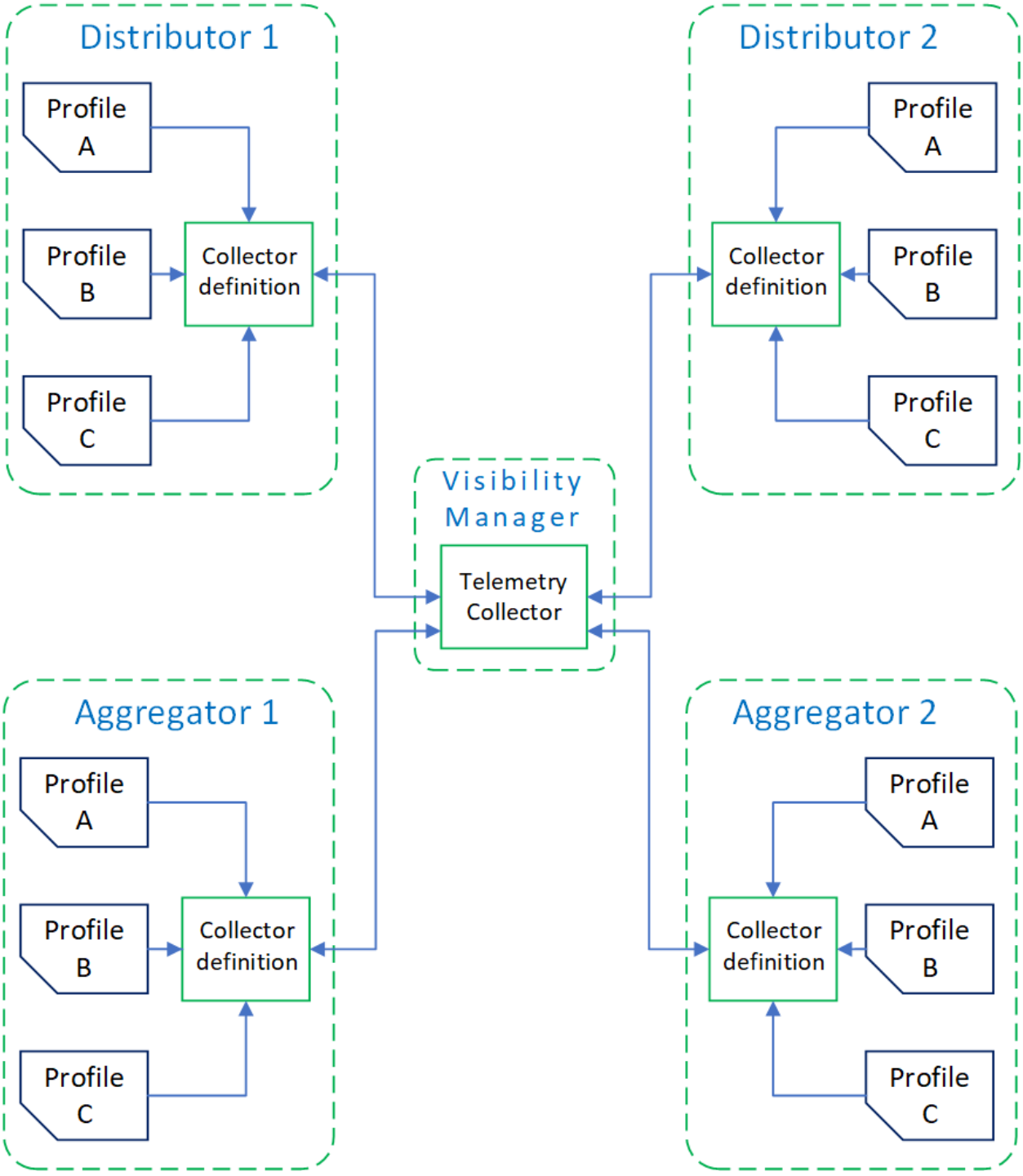


Figure 14: NPB-grid and Visibility Manager telemetry collector

Telemetry profiles for Visibility Manager

Profiles contain the following elements:

- Attributes (usually counters), which you can selectively remove from the profile

- An **interval** value (how often data are sent), which you can modify
- (Most profiles) The interfaces that you want the profile to monitor

The following telemetry profile types and profiles are required for management of NPB-grids by Extreme Visibility Manager.

Note
For other supported profiles, refer to the *Extreme SLX-OS Monitoring Configuration Guide*.

interface

Of the **interface** profile-type, the only profile supported is **default_interface_statistics**. This profile tracks data related to the physical interface. You need to specify monitored interfaces and can modify the default streaming interval (30 seconds).

The fields supported by default for this profile are as follows:

- ethernet and port-channel interfaces
- interval
- In/Out packets
- In/Out unicast packets
- In/Out broadcast packets
- In/Out multicast packets
- In/Out packets per second
- In/Out bandwidth
- In/Out link-utilization
- In/Out octets
- In/Out errors
- In/Out CRC errors
- In/Out discards

pbr

Of the **pbr** profile-type, the only profile supported is **default_pbr_statistics**. This profile tracks policy-based routing (PBR) data. You need to specify monitored interfaces and can modify the default streaming interval (60 seconds).

The fields supported by default for this profile are as follows:

- ethernet and port-channel interfaces
- interval
- acl-byte-count
- acl-hit-count
- acl-name
- acl-seq-num
- if-name
- num-rules
- route-map

system-utilization

Of the **system-utilization** profile-type, the only profile supported is **default_system_utilization_statistics**. This profile tracks system-related data. You can modify the default streaming interval (60 seconds).

The fields supported by default for this profile are as follows:

- Interval
- Total system memory
- Total used memory
- Total free memory
- Cached memory
- Buffers
- Total swap memory
- User free memory
- Kernel free memory
- Total swap memory
- Total free swap memory
- Total used swap memory
- User process
- System process
- Niced process
- In/out wait
- HW interrupt
- SW interrupt
- Idle state
- Steal time
- Up time

event

Of the **event** profile-type, the only profile supported is **default_event_statistics**. This profile tracks up/down interface events.

The fields supported by default for this profile are as follows:

- event-class
- payload
- severity
- timestamp

lldp

Of the **lldp** profile-type, the only profile supported is **default_lldp_statistics**. This profile tracks LLDP neighbor information, including link states. You need to specify monitored interfaces and can modify the default streaming interval (30 seconds).

The fields supported by default for this profile are as follows:

- if-name
- link-state
- local-chassis-id
- local-lag-id
- local-port
- remote-chassis-id
- remote-lag-id
- remote-port
- speed
- update-type

Configuring telemetry profiles for Visibility Manager

Procedure

1. In privileged EXEC mode, enter **configure terminal**.

```
device# configure terminal
```

2. Enter the **telemetry profile** command to configure the profile.

```
device(config)# telemetry profile interface default_interface_statistics
```

3. (For the **interface**, **lldp**, or **pbr** profile types) Enter the **interface** command, specifying the interfaces to monitor.
   - To specify individual interfaces or ranges of interfaces, use the **add** keyword.

```
device(config-interface-default_interface_statistics)# interface ethernet add
0/1-2,0/7
```

   - To specify all interfaces, use the **all** keyword.

```
device(config-interface-default_interface_statistics)# interface port-channel all
```

   - To specify all—with the exception of certain—interfaces, use the **except** keyword.

```
device(config-interface-default_interface_statistics)# interface ethernet except
0/1-2,0/7
```

   - To remove specified interfaces, use the **remove** keyword.

```
device(config-interface-default_interface_statistics)# interface ethernet remove
0/3-4,0/6
```

   - To remove all interfaces, use the **none** keyword.

```
device(config-interface-default_interface_statistics)# interface ethernet none
```

Note

The **interface** and **pbr** profile types enable you to specify port-channels for monitoring—with the option of also specifying Ethernet interfaces. For such multiple support, enter multiple **interface** commands.

4. (For profile types other than **event** and **lldp**) To modify the default interval, enter the **interval** command.

```
device(config-interface-default_interface_statistics)# interval 30
```

5. To remove a default attribute, enter the **no add** command.

```
device(config-system-utilization-default_system_utilization_statistics)# no add buffers
```

6. To restore a default attribute that was previously removed, enter the **add** command.

```
device(config-system-utilization-default_system_utilization_statistics)# add buffers
```

7. To exit configuration mode—saving the configuration—enter **exit**.

```
device(config-interface-default_interface_statistics)# exit
```

Example

The following example specifies the monitored interfaces and changes the default interval.

```
device# configure terminal
device(config)# telemetry profile interface default_interface_statistics
device(config-interface-default_interface_statistics)# interval 30
device(config-interface-default_interface_statistics)# interface ethernet add 0/1-2,0/7
device(config-interface-default_interface_statistics)# exit
```

External-collector streaming for Visibility Manager

On each aggregator and distributor device, you configure a **telemetry collector** object that specifies the following parameters:

- Supported telemetry profiles of the following profile types:
  - **event**
  - **interface**
  - **lldp**
  - **pbr**
  - **system-utilization**
- IPv4 or IPv6 address and port of the Visibility Manager telemetry collector
- Encoding format: JavaScript object notation (JSON)
- Activation

Configuring the telemetry collector for Visibility Manager

Procedure

1. In privileged EXEC mode, enter **configure terminal**.

```
device# configure terminal
```

2. Enter the **telemetry collector** command, specifying a collector name.

```
device(config)# telemetry collector collector_1
```

3. Enter the **ip port** command, specifying the IPv4/IPv6 address and port of the Visibility Manager telemetry collector.

```
device(config-collector-collector_1#) ip 10.168.112.10 port 1
```

4. Enter the **profile** command to include the five telemetry profiles that you configured.

```
device(config-collector-collector_1)# profile system-profile
default_system_utilization_statistics
device(config-collector-collector_1)# profile interface-profile
default_interface_statistics
device(config-collector-collector_1)# profile event-profile default_event_statistics
device(config-collector-collector_1)# profile lldp-profile default_lldp_statistics
device(config-collector-collector_1)# profile pbr-profile default_pbr_statistics
```

5. Enter **encoding json** to specify the JSON encoding format.

```
device(config-collector-collector_1)# encoding json
```

6. Enter the **activate** command to activate the collector.

```
device(config-collector-collector_1)# activate
```

7. To confirm the configuration, enter the **show running-configuration telemetry collector** command.

```
device(config-collector-collector_1)# do show running-config telemetry collector
collector_1
telemetry collector collector_1
ip 192.0.2.61 port 54322
ipv6 2001:db8:: port 54322
profile system-profile default_system_utilization_statistics
profile interface-profile default_interface_statistics
profile lldp-profile default_lldp_statistics
profile event-profile default_event_statistics
profile pbr-profile default_pbr_statistics
encoding json
activate
!
```

8. To display the status of a telemetry collector, enter the **show telemetry collector** command.

```
device(config-collector-collector_1)# do show telemetry collector name collector_1
Telemetry data is being streamed to  <collector_1>  on 192.0.2.61:54322

Profiles Streamed            Interval Uptime<DD/HH:MM:SS> Last Streamed
---------------------------- -------- ------------------- -------------
default_event_statistics     0 sec    0/19:55:11          2019-03-21 13:32:05

default_interface_statistics 30 sec   0/19:55:11          2019-03-21 22:46:45

default_lldp_statistics      30 sec   0/19:55:11          2019-03-21 13:32:08

default_pbr_statistics       240 sec  0/19:55:11          2019-03-21 22:43:44

default_system_utilization   60 sec   0/19:55:11          2019-03-21 22:46:45
_statistics
!
```

9. To display the status of active telemetry collector sessions, enter the **show telemetry collector summary** command.

```
device# show telemetry collector summary
Activated Collectors:
--------------------
Name         IP Address:Port   Streaming/Connection Status
----------   ---------------   ---------------------------
collector_1  192.0.2.61:54322  streaming
```

Example

The following example configures and activates a telemetry collector.

```
device# configure terminal
device(config)# telemetry collector collector_1
device(config-collector-collector_1)# profile system-profile
default_system_utilization_statistics
device(config-collector-collector_1)# profile interface-profile
default_interface_statistics
device(config-collector-collector_1)# profile event-profile default_event_statistics
device(config-collector-collector_1)# profile lldp default_lldp_statistics
device(config-collector-collector_1)# profile pbr default_pbr_statistics
device(config-collector-collector_1)# ip 10.168.112.10 port 1
device(config-collector-collector_1)# encoding json
device(config-collector-collector_1)# activate
```

TAP configurations

Your options are as follows:

- Forwarding NPB-grid traffic on page 74
- Replicating traffic to multiple NPB-grid destinations on page 75
- Aggregating traffic from TAPs on page 77

Forwarding NPB-grid traffic

About This Task

Note
This task assumes a one-destination destination-group.

Procedure

1. Configure a regular ACL or a user-defined ACL (UDA):
   - For a regular ACL, refer to Creating ACLs for NPB on page 11.
     ```
     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 12.
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 target destination-group.

   Note
   For load balancing, specify a destination group that contains one or more load-balance groups.

   - Without header modification:
     ```
     device(config-route-map-npb_map1/permit/1)# set interface pbf-destination-group 9
     ```
   - With header modification:
     ```
     device(config-route-map-npb_map1/permit/1)# set interface pbf-destination-group 9
     strip-vlan outer
     ```
4. Exit to global configuration mode.
   ```
   device(config-route-map-npb_map1/permit/1)# exit
   ```
5. Apply the route map to the TAP.
   - 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
     ```

Example

After configuring a destination group with one destination, the following example forwards ingress traffic from Ethernet 0/1 to the destination.

```
device# configure terminal
device(config)# pbf destination 1
device(config-destination-1)# precedence 10 set interface port-channel 1
device(config-destination-1)# exit
device(config)# pbf destination-group 9
device(config-destination-group-9)# add destination 1
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 pbf-destination-group 99
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
```

Replicating traffic to multiple NPB-grid destinations

Before You Begin

Note
For the single-NPB implementation, refer to Replication to multiple interfaces (single-NPB) on page 29.

Procedure

1. Configure a regular ACL or a user-defined ACL (UDA):
   - For a regular ACL, refer to Creating ACLs for NPB on page 11.
     ```
     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 12.
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 target multiple-destination destination-group.

   Note
   For load balancing, specify a destination group that contains one or more load-balance groups.

   - Without header modification:
     ```
     device(config-route-map-npb_map1/permit/1)# set interface pbf-destination-group 99
     ```
   - With header modification:
     ```
     device(config-route-map-npb_map1/permit/1)# set interface pbf-destination-group 99
     strip-vlan outer
     ```
4. Exit to global configuration mode.
   ```
   device(config-route-map-npb_map1/permit/1)# exit
   ```
5. Apply the route map to the TAP interface.
   ```
   device(config)# interface ethernet 0/2
   device(conf-if-eth-0/2)# npb policy route-map npb_map1
   ```

Example

After configuring destinations and a destination group, the following example forwards ingress traffic from Ethernet 0/1 to the destinations contained in destination-group 99.

```
device# configure terminal
device(config)# pbf destination 10
device(config-destination-10)# precedence 10 set interface port-channel 1
device(config-destination-10)# exit
device(config)# pbf destination 20
device(config-destination-20)# precedence 10 set interface ethernet 0/20
device(config-destination-20)# exit
device(config)# pbf destination 30
device(config-destination-30)# precedence 10 set interface port-channel 3
device(config-destination-30)# exit
device(config)# pbf destination-group 99
device(config-destination-group-99)# add destination 10,20,30
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 pbf-destination-group 99
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
```

Aggregating traffic from TAPs

Before You Begin

Note
For the single-NPB implementation, refer to Aggregating traffic from interfaces (single-NPB) on page 28.

Procedure

1. Configure a regular ACL or a user-defined ACL (UDA):
   - For a regular ACL, refer to Creating ACLs for NPB on page 11.
     ```
     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 12.
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 target destination-group.

   Note
   For load balancing, specify a destination group that contains one or more load-balance groups.

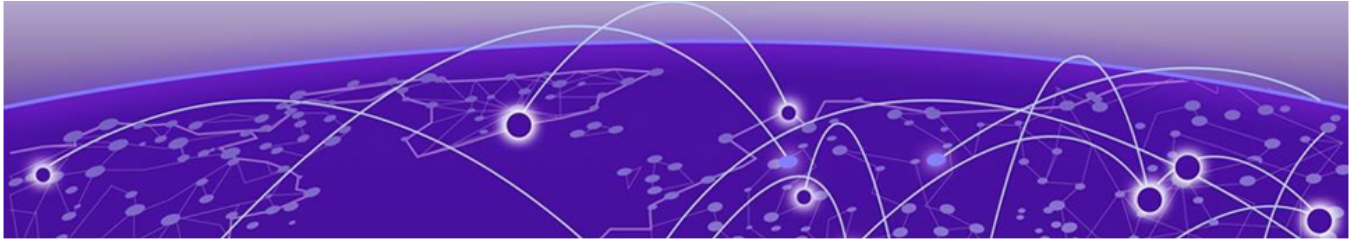
   - Without header modification:
     ```
     device(config-route-map-npb_map1/permit/1)# set interface pbf-destination-group 9
     ```
   - With header modification:
     ```
     device(config-route-map-npb_map1/permit/1)# set interface pbf-destination-group 9
     strip-vlan outer
     ```
4. Exit to global configuration mode.
   ```
   device(config-route-map-npb_map1/permit/1)# exit
   ```
5. Apply the route map to the TAPs.
   - 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
     ```

Example

After configuring a destination group with one destination, the following example forwards ingress traffic from Ethernet 0/1 and port-channel 100 to the destination.

```
device# configure terminal
device(config)# pbf destination 1
device(config-destination-1)# precedence 10 set interface port-channel 1
device(config-destination-1)# exit
device(config)# pbf destination-group 9
device(config-destination-group-9)# add destination 1
```

```
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 pbf-destination-group 99
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
```



VXLAN Transit and Native Mode

Introduction

Generally, tunneled flows are only filtered on inner headers when the outer headers are stripped at the ingress interface. Otherwise, the outer headers are used for filtering.

The new VXLAN filtering modes introduce the ability to filter tunneled flows on their inner headers even if the outer headers are not stripped at the ingress interface. VXLAN Transit mode enables you to apply Layer 2 and Layer 3 policies on the inner headers where the outer headers are not stripped.

In the VXLAN Native mode, only the outer native headers (L2/L3/L4) are parsed. Parsing stops at L4 headers and any tunnel headers following the L4 headers are not parsed.

Another feature of these modes is the ability to use the complete 128-bit IPv6 SIP/DIP for filtering the traffic on the interface. Other modes only enable filtering on the lower or upper 64-bits of SIP/DIP.

Parser Profile

VXLAN modes are implemented with a new parser profile. A new parser profile was implemented as the existing profile has exhausted some TCAM resources. Since the memory available for TCAM is limited, the new parser profile does not support the complete set of flows that the default parser supports. The following table lists the options supported by the two parser profiles.

Profile	Supported Flows
Default	Supports GTP, VXLAN, GRE, MPLS, IP-IP, Partial-IPv6 (64b SIP/DIP)
Vxlan-Transit-128b-V6IP	Supports GTP, VXLAN, GRE, IP-IP, VXLAN-Transit, Full-IPv6 (128b SIP/DIP)

You must use the *Vxlan-Tranist-128b-V6IP* parser profile for VXLAN filtering modes.

Profile Usage

Keep the following guidelines in mind when switching from *Default* profile to *Vxlan-Transit-128b-V6IP* profile:

- No interface on the device must be configured with *strip-mpls*.
- There should be no UDA profiles that match *MPLS* flows. These flows are generally defined using the header-type MPLS.

Keep the following guidelines in mind when switching from *Vxlan-Transit-128b-V6IP* profile to *Default* profile:

- No interface on the device must be configured with either *VXLAN Transit mode* or *VXLAN Native mode*.
- There should be no UDA profiles that match *IPV6_128* header flows.

You will not be allowed to switch between profiles if the above conditions are not met. You will only be allowed to switch profiles after removing the configurations which are considered invalid under the new desired profile.

VXLAN Transit Mode

The VXLAN Transit mode enables you to filter traffic on the VXLAN inner headers even when the outer headers are not stripped. On non VXLAN Transit mode interfaces, tunnel flows are filtered based on the inner headers only when the outer headers are stripped.

In the VXLAN Transit mode, only VXLAN traffic is processed. All other traffic types will be considered Ethernet traffic with Unassigned L3 Protocol. For VXLAN traffic ingressing on an interface configured in this mode, L2/L3 policies are always applied on the inner headers irrespective of whether outer header stripping is enabled or disabled.

The following are the limitations and other things to consider when using the VXLAN Transit mode:

- The interface in VXLAN Transit mode cannot parse any other traffic other than VXLAN. Any non-VXLAN traffic is treated as an Ethernet frame with unknown L3 protocol.
- You cannot use any interface-level editing configurations for an interface in this mode. The only permitted configuration is *strip-vxlan*, which is used to strip the external headers for traffic ingressing on this interface.
- The switch should be running *Vxlan-Transit-128b-V6IP* profile.
- The selected interface cannot have any *allow-vn-tag* configurations.
- The selected interface cannot have any strip commands configured on it. The only allowed strip configuration is *strip-vxlan*.
- The selected interface cannot have any route-maps which define VLAN add/delete operations on the egress interface.
- The selected interface cannot have an UDA profile with header-type *IPV6* applied on it.
- The selected interface cannot have an IPv6 ACL applied on it through a route-map.
- You cannot disable VXLAN Transit mode on the interface if it has a UDA profile with the header-type *IPV6_128* applied on it.
- You cannot disable VXLAN Transit mode on the interface if it has an IPv6 ACL configured on it.

Enable or Disable VXLAN Transit Mode

Use the following commands to enable or disable VXLAN Transit mode on an interface.

```
mode vxlan-transit
no mode vxlan-transit
```

Defining UDA Profiles

VXLAN traffic on a VXLAN Transit mode interface is parsed differently when compared to VXLAN traffic on a regular interface. A flow definition for a VXLAN Transit mode interface will be different from a flow definition for a regular interface.

Flow definition for a normal interface:

```
SLX(config)# uda-key profile UDA_PROFILE1
SLX(config-uda-key)# flow header0 ETHERNET header1 IPV4 header2 UDP header3 VXLAN header4
TUN_ETHERNET header5 IPV4 header6 TCP header7 PAYLOAD4
```

Flow definition for a VXLAN Transit mode interface:

```
SLX(config)# uda-key profile UDA_PROFILE2
SLX(config-uda-key)# flow header0 ETHERNET header1 IPV4 header2 TCP header3 PAYLOAD32
```

Flow definition when parsing a IPv6 inner header:

```
SLX(config)# uda-key profile UDA_PROFILE2
SLX(config-uda-key)# flow header0 ETHERNET header1 IPV6_128 header2 TCP header3 PAYLOAD32
```

Note

Payloads are not matched when using UDA on an interface in the VXLAN Transit mode. You can still select payload fields using CLI, however, these fields will not be matched during parsing.

Native Mode

Use the *Native* mode if you want to parse the complete 128b SIP/DIP headers. In this mode, only the native headers (L2/L3/L4 headers) are parsed. Parsing stops after the L4 headers. If any tunnel headers are encountered, they are not parsed.

In the Native mode, only native IP (v4 or v6) and L4 headers are parsed. This mode does not parse any tunnel traffic.

The following are the limitations and other things to consider when using the VXLAN Native mode.

- The interface in VXLAN Native mode cannot parse Tunneled traffic.
- The interface can only process Ethernet headers that can be either *Untagged* or *C-Tagged*. Any other tag combination will not be parsed and such a frame will be classified as an Ethernet frame with unknown L3 protocol.

- The selected interface cannot have any interface-level editing configuration such as *strip-VXLAN*, *strip-nvgre*, *strip-mpls* or similar.
- The selected interface cannot have *allow-vn-tag* configured.
- The selected interface cannot have an UDA profile with header-type *IPV6* applied on it.
- The selected interface cannot have an IPv6 ACL applied on it through a route-map.
- You cannot disable VXLAN Native mode on the interface if it has a UDA profile with the header-type *IPV6_128* applied on it.
- You cannot disable VXLAN Native mode on the interface if it has an IPv6 ACL configured on it.

Enable or Disable VXLAN Native Mode

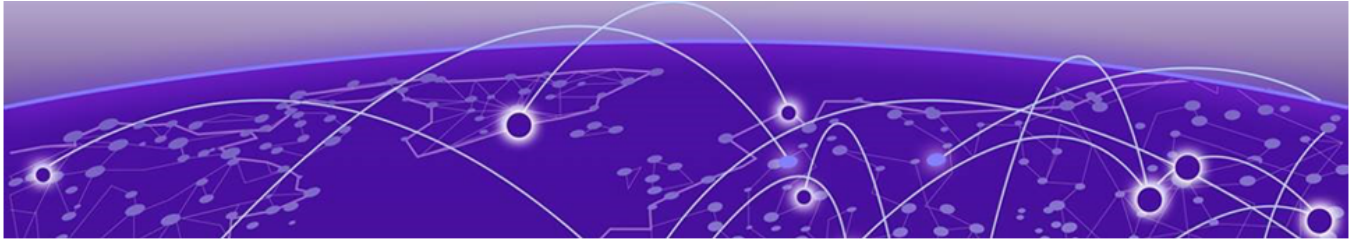
Use the following commands to enable or disable VXLAN Native mode on an interface.

```
mode native-ip
no mode native-ip
```

Defining UDA Profiles

The UDA definition is similar to the configuring flow definitions on a normal interface. However, since this mode support parsing the complete 128b SIP/DIP, the flow definition to parse this header is different. If you want to use the complete 128b IPv6 SIP/DIP address, use the *IPV6_128* definition in your UDA.

```
SLX(config)# uda-key profile UDA_PROFILE1
SLX(config-uda-key)# flow header0 ETHERNET header1 IPV6_128 header2 TCP header3 PAYLOAD32
```



NPB show commands

Table 33: 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 pbf destination	Displays information about one PBF destination or about all PBF destinations.
show pbf destination-group	Displays information about one PBF destination-group or about all PBF destination-groups.
show route-map	Displays information of all route maps, of a specific route map, or of the route map applied to an interface.
show running-config pbf destination	Displays configuration details of one PBF destination or of all PBF destinations.
show running-config pbf destination-group	Displays configuration details of one PBF destination-group or of all PBF destination-groups.
show running-config tvf-domain	Displays configuration details of one or all TVF domains.