Extreme 9920 Software Configuration Guide, 21.1.0.0

Supporting Extreme 9920

9037100-00 Rev AA
June 2021

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/

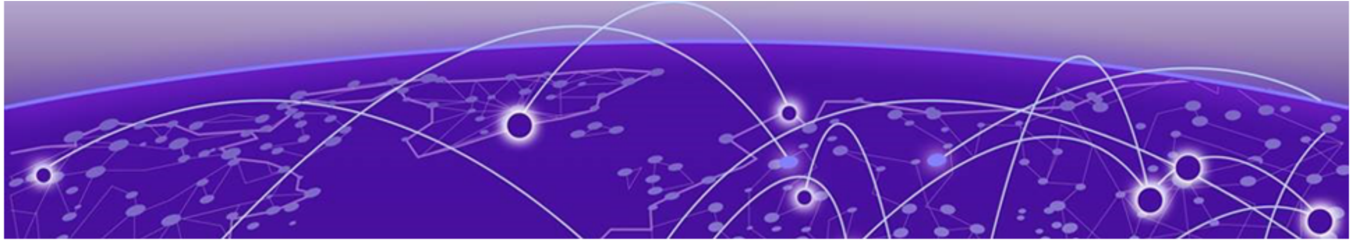
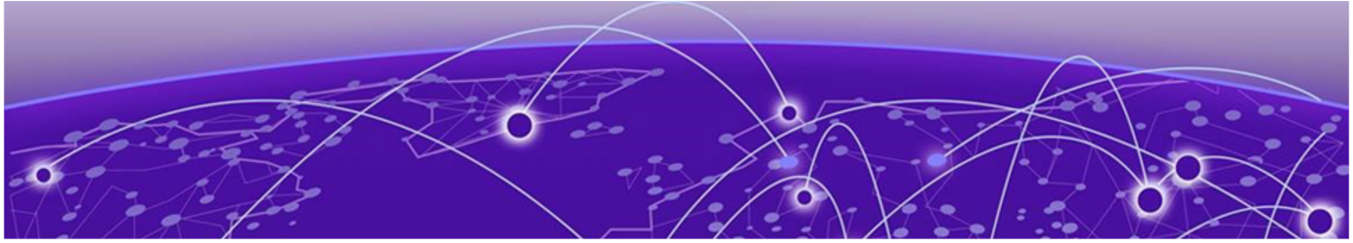


Table of Contents



Preface

Read the following topics to learn about:

- The meanings of text formats used in this document.
- Where you can find additional information and help.
- How to reach us with questions and comments.

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
⚠	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 is displayed 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 Product Announcements

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

1. Go to The Hub.
2. In the list of categories, expand the **Product Announcements** list.
3. Select a product for which you would like to receive notifications.
4. Select **Subscribe**.
5. To select additional products, return to the **Product Announcements** list and repeat steps 3 and 4.

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

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:

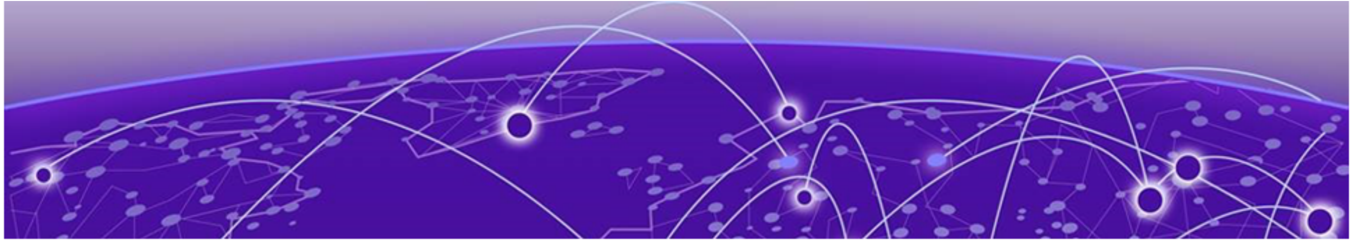
- Content errors, or confusing or conflicting information.

- 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.

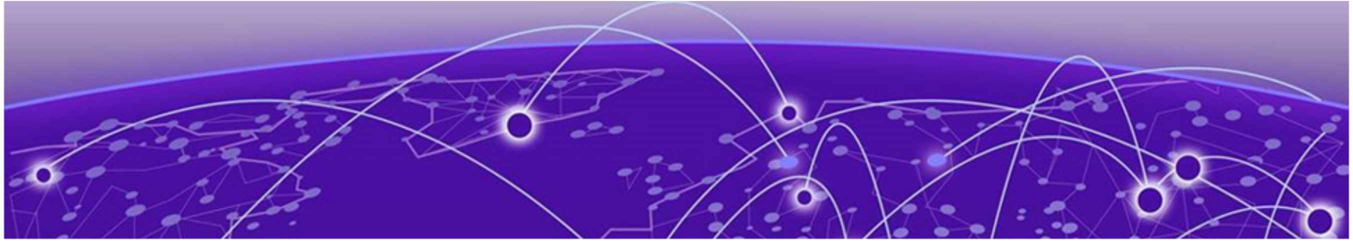
Provide the publication title, part number, and as much detail as possible, including the topic heading and page number if applicable, as well as your suggestions for improvement.



What's New in this Document

This document is new for the release of the Extreme 9920 software with the NPB application.

For more information about this release, see the *Extreme 9920 Software Release Notes, 21.1.0.0*.



Introduction to the NPB Application

The Extreme 9920 device runs an operating system (TierraOS) that runs one or more applications. This version of the Extreme 9920 software has one application (NPB application).

The NPB application provides network packet broker functions:

- Aggregate, inspect, and classify network traffic from ingress ports
- Process network traffic according to policies
- Forward traffic to analytic applications

The following topics introduce the application and supported devices.

NPB Application Overview

The NPB application provides functionality to process and prepare packets for visibility tools. This allows core networking devices to offload network monitoring.

When the Extreme 9920 running Extreme 9920 software is attached to optical taps between the core networking devices, a copy of the traffic is sent to the Extreme 9920 for filtering traffic of interest and formatting before being sent on to visibility tools.

The NPB application supports the following features:

- Aggregation: Aggregates traffic arriving from multiple ports and directs it to a single port or port-channel ("many to one").
- Replication: Replicates network traffic to multiple ports and port-channels ("one to many").
- Load balancing: Distributes network traffic among ports in a port-channel.
- ACL filtering: Directs network traffic based on Layer 2 to Layer 4 protocol headers.
- Route-map forwarding: Redirects packets based on Layer 2 to Layer 4 Protocol headers to the desired physical port or port-channel interfaces.
- Packet slicing: Truncates length of the packet to specified length.
- Tunnel origination: Encapsulates packets with IPv4 Generic Routing Encapsulation (GRE) headers.
- Tunnel termination: Tunnel termination classifies and decapsulates incoming IPv4 packets.
- Encapsulation-header stripping:
    - Removes tags that are not supported by visibility applications.
    - Supports 802.1BR, VN-Tag, VLAN, VXLAN, GTPU, GRE, and IPIP headers.

Supported Device Information

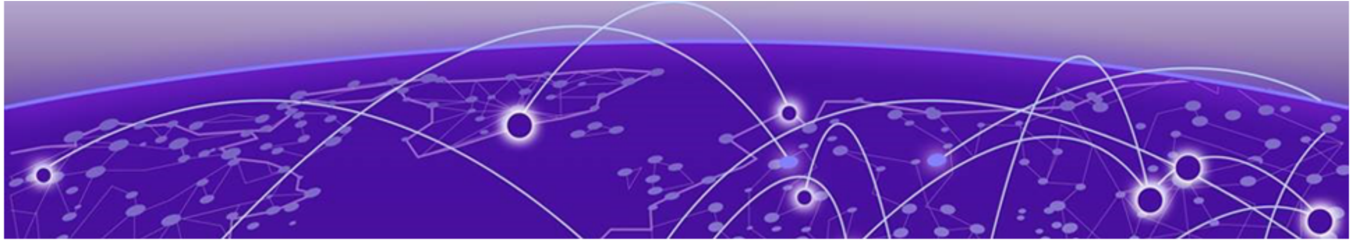
The Extreme 9920 software with the NPB application runs on Extreme 9920 devices.

Default CA certificate

A default CA certificate for TLS is provided to verify the certificate that the Extreme 9920 device issues. All gNMI requests to the device are over a secure channel. The Extreme 9920 uses the default HTTPS server certificate if you do not import an HTTPS server certificate.

Use the following CA certificate on the client to verify the certificate generated by the 9920.

```
-----BEGIN CERTIFICATE-----
MIIGPjCCBCagAwIBAgICEAAwDQYJKoZIhvcNAQELBQAwgbMxCzAJBgNVBAYTAlVT
MQswCQYDVQQIDAJDQTELMAkGA1UEBwwCU0oxGTAXBgNVBAoMEEV4dHJlbWUgTmV0
d29ya3MxHzAdBgNVBAsMFkV4dHJlbWUgTmV0d29ya3MgTkdOUEIxIjAgBgNVBAMM
GW5nbnBiLmV4dHJlbWVuZXR3b3Jrcy5jb20xKjAoBgkqhkiG9w0BCQEWG3N1cHBv
cnRAZXh0cmVtZW5ldHdvcmtzLmNvbTAeFw0yMDA3MDcyMTEzNDNaFw0zMDA3MDUy
MTEzNDNaMIGnMQswCQYDVQQGEwJVUzELMAkGA1UECAwCQ0ExGTAXBgNVBAoMEEV4
dHJlbWUgTmV0d29ya3MxJDAiBgNVBAsMG0V4dHJlbWUgTmV0d29ya3MgTmV4dEdl
bk5QQjEeMBwGA1UEAwwVTkdOUEIgSW50ZXJtZWRpYXRlIENBMSowKAYJKoZIhvcN
AQkBFhtzdXBwb3J0QGV4dHJlbWVuZXR3b3Jrcy5jb20wggIiMA0GCSqGSIb3DQEB
AQUAA4ICDwAwggIKAoICAQC64lUkRcEvz+jWfm9V9+g/AgZFpDOKL5oR4c3IHdwM
vAA6Rt+Os+6wvOpLysDvzggeVh4L6BWULgFw5SyRhjKJbyz7PaMBg/id5XPqWntU
3MoPOdewdVozyZZf3MRDVqgw8f7nT4Ex55fSnfyYLOx5g2++rUUK3jpQo74vRI/W
SUZdOAvs9hkERcMJIm4DDcj86Z4HuYaB/iBBSqPDhRoErpaX36TOWY+2wCNomkK1
zzCO9PW3HhfZk+GWF10U/7ZkNOBMnd5nVIalf+VsSpaPzQxAJtIKS1xYqmoACW6s
S/myGEPuDqmYhilwSgP+lyRmpkGZEfPbwxZyrhxUAnwQ0+r8HUSvBRavK+utt+JQ
SeFiPPVeo6OoGgwRJ1t9KVid+Sp56+gwMj27Kf26cWYUJsjjHxyJgFFCforcn40M
Kox5idbZkQjdo7ciofK7Twz8U+ip/lyhbycUcz7cG7vimRvu+BpyJ29zL7It/PZX
U8fP3r0ssudasfwZGxl3AO58szhopE0m4eaIQzhotqwqXt6s8Vh/+qj6JMudMxa6
5HVeBVX6BEvLG8TwKaaQiJ4edwI/QY0WZ0wxfeDn05haSiyOhRkmA/F9cqv3h8qM
B5+IZ+nYNpjASyxc1QTtW9Xhn37vp61+7JddU9zxeSVkB43YGOK9Uq1+DXDawlzA
5wIDAQABo2YwZDAdBgNVHQ4EFgQU2Sn6JG7Jo3QzMRBsLHhQkVW8uzQwHwYDVR0j
BBgwFoAUuwS1PNyE3M192izM7zaoJm1vhTQwEgYDVR0TAQH/BAgwBgEB/wIBADAO
BgNVHQ8BAf8EBAMCAYYwDQYJKoZIhvcNAQELBQADggIBAKRIQfsZiiIVZC3jCmBt
cwf3LRN2ESOy8bj4AV0LxgchjMtw/y/Dp7ST5FkUEIAya8HEmL1tjZhHFh0uNfBx
7UcRV2R7ZhGu08TujFbo9sVy2GHd+w1/L6VDauEjV6eUd4oI6kylcDK/OA3UoYwF
vbiiLlDYEFaP3/3MFaqk9osfmkxcmhu3qxOt3QxqevXPiXtXlNxT4w/LrQFXKMcJ
40zjFjgnNsdYlr92c2kwWDE/44xnOWEH7Ar2PuqvqHJi1GFv1lV/Ys+0wkqCyy/K
eAdde8d8ZWcXroSHzlGI34Tql2U9+bZjxNgU9Nc8VJGq+K9LBDTQqFfGg0n9qHYE
eCGGzw5eT+lJMSVompRPEHt44qCk5eKRmEWsMbgeD8d6cisPE4PffIynSV/evjsY
k9gaE0d86uhN5EuNQsqtLn5vsdWN8nBBP+umPLtHpppHATAGSWgW8WGgSLs8gBwS
IABCVCpV7oePXN0LCfzRTd2JwCmUpqxreGw3cuePDMNimOQvwnaAvLfNxFYSoRtp
wU8VGdAUQJxqkeS4xlkpKbOYGGHlntlskSP7VuygLn2ISa3v6xTRLlaTDcLuiu+K
vNGwM33rgKUiPtDFM9oK0CtiydM1TqfQZB3/B1a3RzqX2OmBvR6qB9M5jeNQXd+T
wa/daP9p2G6/lcNRE+AiCpul
-----END CERTIFICATE-----
```



Network Traffic Monitoring Preparation

To properly configure and use the Extreme 9920 effectively, you need a good understanding of the various components that are used to configure the desired packet workflow to monitor your network.

Before you configure any settings, you need to create components that are used to monitor and modify packets flowing into and out of the 9920. For the purpose of configuration, it is recommended that you adhere to the following configuration order when you create these building blocks. When you have a basic packet workflow in place, modifications to that workflow follow the same order as outlined here.

1. Access-control lists (ACLs)
2. Encapsulations for any tunnel origination
3. Listener-policies
4. Egress
5. Egress-groups
6. Route-maps
7. Interfaces
8. Ingress-groups

It's important to note the terminology used when discussing packet flow through the 9920. Interfaces that are connected to the network being monitored are termed *tap interfaces*. Interfaces connected toward the visibility tools are called *tool ports*. Because tap ports often receive the same traffic types, you create and configure an *ingress-group* to group the tap interfaces to the same desired *route-map*.

Likewise, tool ports are configured in an *egress*, which combines interfaces and any post-processing directives (such as dropping specific packets, removing tunnel or specific headers, adding a VLAN tag, or truncating packet length) by using a listener-policy. In-between the 'ingress-group' and the 'egress' is abstraction layer that connects the two together, the 'egress-group'.

You can configure various actions at each level, and some may depend on a prior entity existing in the configuration. The examples that are provided are not exhaustive for all possible configurations or parameters. For additional information, see *Extreme 9920 Software Command Reference, 21.1.0.0*.

Access Control List Overview

An access Control List (ACL) is a set of rules defined to filter the network traffic. Each ACL is assigned a unique name.

Packet filtering and traffic flow through the network are managed with access control lists (ACLs), which contain rules that you configure for that purpose. Incoming packets are matched against the entries in ACL. Packets are forwarded or dropped based on criteria specified in ACL. The unique sequence number of each entry indicates the order that the packet will be matched against rules in the ACL. The lower the sequence ID, the earlier the rule will be checked against the packet. Care should be taken when designing the ACLs being used to prevent a lower-sequence ID from matching all the traffic desired for a higher sequence ID.

ACLs are classified as MAC (Layer 2), IPV4 (Layer 3), or IPv6 (Layer 3) access list based on the matching keys. If incoming packets match both Layer 2 and Layer 3 ACLs, Layer 3 ACLs in the same route-map stanza are prioritized and actions associated with L3 ACLs are applied.

- ACLs that are referenced in route-maps or listener polices can be modified and deleted.
- When ACL entries that are in use are deleted, the associated route-map or listener-policy re-program the hardware accordingly.
- Each ACL entry can specify its ability to count or log.
    - The counting action provides packet and octet count (64-bit capacity).
    - Logging action sends a copy of the frame to the CPU. The forwarding action remains unchanged.

Create an IP Access Control List

Learn how to create an IP access list, which you can configure with rules that permit or deny traffic based on packet fields belonging to the IPv4 family of protocols.

About This Task

You must observe the following naming conventions when you create an access control list (ACL).

- Name cannot exceed 64 characters.
- Name must start with an alphabetic character or an underscore.

- Name must be an arbitrary sequence of alphabetic characters, numerals, underscores, hyphens, or dots.
- Name must not be a reserved keyword (see the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information).

Procedure

1. Run the **configure terminal** command to access Config mode.

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Create an IPv4 ACL by running the **ip access-list** *acl-name* command .

   ```
   device(config)# ip access-list acl5-ipv4
   device(config-ip-acl)# permit ip any any
   device(config-ip-acl)# end
   device#
   ```

3. Run the following **show** command to confirm the access control list was created.

   ```
   device#show ip access-list all
   ip access-list acl5-ipv4
     seq 10 permit ip any any
   ( 0 Packets, 0 Bytes, 0 Packets/sec, 0 Bits/sec )
   ```

Configure an IPv6 Access Control List

Learn how to create an IPv6 access list, which you can configure with rules that permit or deny traffic based on packet fields of the IPv6 family of protocols.

About This Task

You must observe the following naming conventions when you create an access control list (ACL).

- Name cannot exceed 64 characters.
- Name must start with an alphabetic character or an underscore.
- Name must be an arbitrary sequence of alphabetic characters, numerals, underscores, hyphens, or dots.
- Name must not be a reserved keyword (see the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information).

Procedure

1. Run the **configure terminal** command to access Config mode.

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Create an IPv6 ACL by running the **ipv6 access-list** *acl-name* command .

   ```
   device(config)# #ipv6 access-list P6
   device(config-ipv6-acl)# permit ipv6 any any vlan 500
   device(config-ipv6-acl)# end
   device#
   ```

3. Run the following **show** command to confirm that the access control list was created.

   ```
   device#show ipv6 access-list all
   ipv6 access-list P6
      seq 10 permit ipv6 any any vlan 500
   ( 0 Packets, 0 Bytes, 0 Packets/sec, 0 Bits/sec )
   ```

Create a MAC Access Control List

Learn how to create a MAC access control list, which you can configure with rules that permit or deny traffic based on packet fields of the L2 OSI layer.

About This Task

You must observe the following naming conventions when you create an access control list (ACL).

- Name cannot exceed 64 characters.
- Name must start with an alphabetic character or an underscore.
- Name must be an arbitrary sequence of alphabetic characters, numerals, underscores, hyphens, or dots.
- Name must not be a reserved keyword (see the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information).

Procedure

1. Run the **`configure terminal`** command to access Config mode.

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Create a MAC ACL by running the **`mac access-list`** *`acl-name`* command .

   ```
   device(config)# mac access-list L2
   device(config-mac-acl)# permit any any vlan 350
   device(config-mac-acl)# end
   device#
   ```

3. Run the following **`show`** command to confirm the access control list was created.

   ```
   device# show mac access-list all
   mac access-list L2
     seq 10 permit any any vlan 350
   ( 0 Packets, 0 Bytes, 0 Packets/sec, 0 Bits/sec )
   ```

Encapsulation Overview

Learn about encapsulating packets with Generic Routing Encapsulation (GRE) headers. GRE provides a private secure path for transporting packets.

You can configure the NPB application to encapsulate outgoing packets with tunnel and L2/L3 parameters. The following L2/L3 parameter configurations are required for GRE encapsulation:

- Outer destination MAC address (mandatory)
- Outer source MAC address (mandatory)
- Outer VLAN Tag (optional)
- Outer VLAN PCP (optional)
- Outer Source IPv4 address (mandatory)
- Outer Destination IPv4 address (mandatory)

Note

- Only IPv4 GRE encapsulation is supported in NPB application.
- All flags in GRE encapsulation header are set to 0.

Configure Encapsulation

About This Task

The L2/L3 parameters are configured as part of encap object and the encap object is attached to egress object to encapsulate packets.

Procedure

1. Enable encapsulation.

   ```
   # encap encap-name
   ```

2. Configure the encapsulation type.

   ```
   # encap-type [ gre | erspan ]
   ```

3. Configure the source IPv4 or MAC address.

   - Source IPv4 address:

   ```
   # source-ipv4-addr ip-addr
   ```

   - Source MAC address:

     ```
     # source-mac-addr mac-addr
     ```

4. Configure the destination IPv4 or MAC address.

   - Destination IPv4 address:

   ```
   # destination-ipv4-addr ip-addr
   ```

   - Destination MAC address:

     ```
     # destination-mac-addr mac-addr
     ```

5. Configure the VLAN ID.

   ```
   # vlan-id vlan-id-value
   ```

6. Configure the VLAN priority (PCP) value.

   ```
   # vlan-pcp vlan-pcp-value
   ```

Example

```
encap encap1
 source-mac-addr 00:00:00:11:11:11
 destination-mac-addr 00:00:00:22:22:22
 source-ip-addr 1.1.1.1
 destination-ip-addr 2.2.2.2
 vlan-id 100
 vlan-pcp 3

interface ethernet 1/1
  no  shutdown

interface ethernet 1/2
  no shutdown

interface ethernet 1/1
  ingress-group ingg1

egress eg1
set encap encap1
  precedence 10  interface ethernet 1/2
quit

set egress-group egg1
```

```
    egress eg1
quit

ip access-list v4
 seq 10 permit ip any any
quit

route-map rm1 10
  match ip access-list v4
  egress-group egg1
quit

ingress-group ingg1
  route-map rm1
quit
```

Listener Policy Overview

Learn about listener-policies and how they are used.

The NPB application supports multiple listener policies, each uniquely identified by a listener policy name. Each listener policy can have multiple stanzas, which are uniquely identified by the sequence ID. A listener policy applies an ACL to an egress and the interfaces associated with it, and defines the actions for the matching ACL. When a listener policy is bound to the egress, it applies the listener policy actions to packet flow.

Listener policies support the following actions:

- Permit/deny traffic
- Packet truncation
- Tunnel termination
- Tag stripping
- VLAN tag addition

Create a Listener Policy

Learn how to create a listener-policy, which is used by an egress to combine interfaces and perform post-processing actions.

About This Task

You must observe the following naming conventions when you create a listener policy.

- Name cannot exceed 64 characters.
- Name must start with an alphabetic character or an underscore.
- Name must be an arbitrary sequence of alphabetic characters, numerals, underscores, hyphens, or dots.
- Name must not be a reserved keyword (see the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information).

Note
Sequence ID is required when you create a listener policy; it defines the order in which it is to be processed (see the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information).

Procedure

1. Run the **configure terminal** command to access Config mode.

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Create a listener policy by running the command **listener-policy** *name seq*.

   ```
   device(config)#listener-policy lpl 10
   device(config-listener-policy)# match ip access-list v4_jumbo_traffic
   device(config-listener-policy)# truncate 512
   device(config-listener-policy)# end
   device#
   ```

3. Run the following **show** command to confirm the listener policy was created.

   ```
   device# show listener-policy lp1
   show listener-policy lpl
   listener-policy lpl 10
   forward-action permit
   match ip access-list v4_jumbo_traffic (pending)
   truncate 512
   Policy matches: 0  packets, 0  bytes, 0 Packets/sec, 0 Bits/sec
   ```

Egress Overview

You create an egress to combine it with various policies to perform actions on packets at egress

Tool ports are entered into an egress, which can use a listener-policy to apply some additional processing actions to packets leaving the interfaces of the egress.

Egress Policies

You configure an egress policy to specify actions to be performed on packets that are exiting the 9920.

An egress policy is used to specify actions to be taken at egress on packets exiting 9920 to prepare them for downstream tools. You must configure a match ACL and a listener policy, at a minimum.

Create an Egress

Learn how to define tool ports (egress packet flow) in an egress.

About This Task

You must observe the following naming conventions when you create an egress.

- Name cannot exceed 64 characters.
- Name must start with an alphabetic character or an underscore.
- Name must be an arbitrary sequence of alphabetic characters, numerals, underscores, hyphens, or dots.
- Name must not be a reserved keyword (see the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information).

Procedure

1. Run the **configure terminal** command to access Config mode.

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Create an egress by running the command **egress** *name* .

```
device(config)# egress egress-one
device(config-egress)# precedence 10 interface ethernet 1/10
device(config-egress)# end
device#
```

3. Run the following **show** command to confirm that the egress was created.

```
device# show egress egress-one
     Name : Egress egress-one
     Precedence 10 interface ethernet 1/10
```

Egress-Group Overview

An egress-group is a mechanism that connects an ingress-group and the egress to define how traffic is forwarded to end devices.

Create an Egress Group

Learn how to create an egress group, which you will use to specify where the packets from the 9920 tap interfaces are to be copied for the various tools.

About This Task

You must observe the following naming conventions when you create an egress group.

- Name cannot exceed 64 characters.
- Name must start with an alphabetic character or an underscore.
- Name must be an arbitrary sequence of alphabetic characters, numerals, underscores, hyphens, or dots.
- Name must not be a reserved keyword (see the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information).

When you create an egress-group, you associate it with an egress so that the actions you specify in various policies are performed on traffic at those tool ports.

Procedure

1. Run the **configure terminal** command to access Config mode.

   The command line changes to configuration mode.

```
device(config)#
```

2. Create an egress group by running the command **egress-group** *group-name* .

```
device(config)# egress-group egg123
device(config-egress-group)# set egress egress-one
device(config-egress-group)# end
device#
```

3. Run the following **show** command to confirm that the egress-group was created.

```
device# show running-config egress-group
  egress-group egg123
  egress-one
```

Route-Map Overview

A route map evaluates incoming packets and determines the traffic of interest for the tools based on the ACLs used. Some additional processing of the packets also can be done if needed. Any additional processing by the route map is reflected on all of the egress groups used by that route map.

Create a Route Map

Learn how to create a route-map, which will be used to evaluate packets and specify actions to be performed on them.

About This Task

You must observe the following naming conventions when you create a route map.

- Name cannot exceed 64 characters.
- Name must start with an alphabetic character or an underscore.
- Name must be an arbitrary sequence of alphabetic characters, numerals, underscores, hyphens, or dots.
- Name must not be a reserved keyword (see the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information).

Note
Sequence ID is required when you create a route-map; it defines the order in which it is to be processed (see the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information).

When you create a route map, you associate it with an egress-group using the **set** command.

Procedure

1. Run the **configure terminal** command to access Config mode.

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Create a route map by running the command **route-map** *map-name seq*.

   ```
   device(config)# route-map noc_traffic 10
   device(config-route-map)# match ip access-list v4_external
   device(config-route-map)# set egress-group netscout
   device(config-route-map)# end
   device#
   ```

3. Run the following **show** command to confirm that the egress was created.

   ```
   device# show route-map all
     rmap1 10
     egress-group egg123
   ```

Interface Overview

Extreme 9920 interfaces can operate in various modes, but the primary use case is to enable them for use, then apply the ingress-group that will match the traffic of interest coming from the monitored network.

Create an Interface

Learn how to create an interface or a range of interfaces.

About This Task

Interface name must be in slot/port format. For more information, see the *Extreme 9920 Software Command Reference, 21.1.0.0*.

When you create an interface, you associate it with an ingress group so that the actions you specify in various policies are performed on traffic at those tool ports.

Procedure

1. Run the **configure terminal** command to access Config mode.

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Create an interface or interface range by running the command **interface** *slot / port-number*.

   ```
   device(config)# interface e 1/10-14,3/3-4
   device(config-if-eth-1/10-14,3/3-4)# no shutdown
   device(config-if-eth-1/10-14,3/3-4)# set ingress-group noc_traffic
   device(config-if-eth-1/10-14,3/3-4)# end
   ```

3. Run the following **show** command to confirm that the interface configuration was accepted.

   ```
   device# show ingress-group noc_traffic

       Name : noc_traffic
          Route-Map : -
        Description : -
         Interfaces :  ethernet 1/10  ethernet 1/11
           ➥ethernet 1/12  ethernet 1/13  ethernet 1/14
           ➥ethernet 3/3  ethernet 3/4
   ```

4. Run the following **show** command to view status for the interfaces you configured.

   ```
   device# show interface ethernet 1/10-14,3/3-4 | include state
   ethernet 1/10 Admin state UP      Operational state UP
   ethernet 1/11 Admin state UP      Operational state UP
   ethernet 1/12 Admin state UP      Operational state UP
   ethernet 1/13 Admin state UP      Operational state UP
   ethernet 1/14 Admin state UP      Operational state UP
   ethernet 3/3 Admin state UP       Operational state UP
   ethernet 3/4 Admin state UP       Operational state UP
   ```

Ingress-Group Overview

Learn about how an ingress-group is used.

An ingress-group is used to link tap interfaces to a specified route-map, which defines actions to be taken on packets at ingress.

Create an Ingress Group

Create an ingress group to classify packets received on the 9920 at ingress.

About This Task

You must observe the following naming conventions when you create an ingress group.

- Name cannot exceed 64 characters.
- Name must start with an alphabetic character or an underscore.
- Name must be an arbitrary sequence of alphabetic characters, numerals, underscores, hyphens, or dots.
- Name must not be a reserved keyword (see the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information).

When you create an ingress-group, you associate it with a route-map using the set command so that the actions you specify in various policies are performed on traffic at the specified tap interface.

Procedure

1. Run the **configure terminal** command to access Config mode.

   The command line changes to configuration mode.

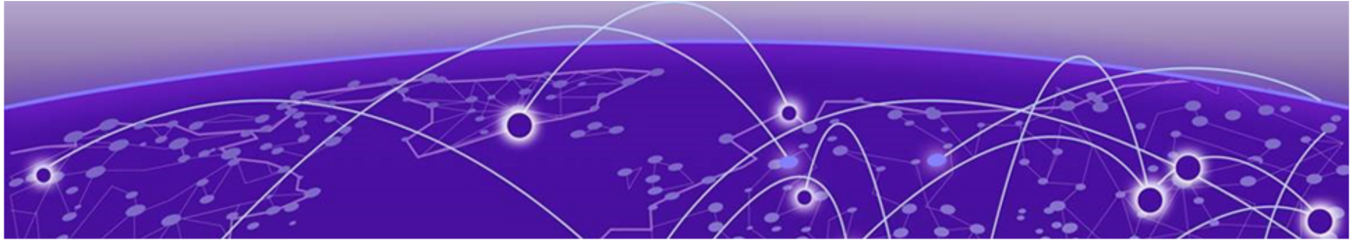
   ```
   device(config)#
   ```

2. Create an ingress group by running the command **ingress-group** *group-name*.

   ```
   device# configure terminal
   device(config)# ingress-group ingress-group noc_traffic
   device(config-ingress-group)# set route-map rmap1
   ```

3. Run the following **show** command to confirm that the ingress group was created.

   ```
   device# show running-config ingress-group
   ingress-group noc_traffic
     set route-map rmap1
   ```



Network Traffic Monitoring

Packet forwarding and filtering are core features of Extreme 9920 software. The following topics discuss how to configure packet forwarding and filtering to monitor and modify network traffic. For information about creating the basic components that you use for configuration, see Network Traffic Monitoring Preparation on page 13.

Encapsulation

Learn about encapsulating packets with Generic Routing Encapsulation (GRE) headers. GRE provides a private secure path for transporting packets.

You can configure outgoing-packets encapsulation with configured tunnel and L2/L3 parameters. The following L2/L3 parameter configurations are required for GRE encapsulation:

- Outer destination MAC address (mandatory)
- Outer source MAC address (mandatory)
- Outer VLAN Tag (optional)
- Outer VLAN PCP (optional)
- Outer Source IPv4 address (mandatory)
- Outer Destination IPv4 address (mandatory)

Note

- Only IPv4 GRE encapsulation is supported in NPB application.
- All flags in GRE encapsulation header are set to 0.

Configure Encapsulation

About This Task

The L2/L3 parameters are configured as part of encap object and the encap object is attached to egress object to encapsulate packets.

Procedure

1. Enable encapsulation.

   ```
   # encap encap-name
   ```

2. Configure the encapsulation type.

   ```
   # encap-type [ gre | erspan ]
   ```

3. Configure the source IPv4 or MAC address.

   - Source IPv4 address:

   ```
   # source-ipv4-addr ip-addr
   ```

   - Source MAC address:

     ```
     # source-mac-addr mac-addr
     ```

4. Configure the destination IPv4 or MAC address.

   - Destination IPv4 address:

   ```
   # destination-ipv4-addr ip-addr
   ```

   - Destination MAC address:

     ```
     # destination-mac-addr mac-addr
     ```

5. Configure the VLAN ID.

   ```
   # vlan-id vlan-id-value
   ```

6. Configure the VLAN priority (PCP) value.

   ```
   # vlan-pcp vlan-pcp-value
   ```

Example

```
encap encap1
 source-mac-addr 00:00:00:11:11:11
 destination-mac-addr 00:00:00:22:22:22
 source-ip-addr 1.1.1.1
 destination-ip-addr 2.2.2.2
 vlan-id 100
 vlan-pcp 3

interface ethernet 1/1
  no  shutdown

interface ethernet 1/2
  no shutdown

interface ethernet 1/1
  ingress-group ingg1

egress eg1
set encap encap1
  precedence 10  interface ethernet 1/2
quit

set egress-group egg1
```

```
    egress eg1
quit

ip access-list v4
 seq 10 permit ip any any
quit

route-map rm1 10
  match ip access-list v4
  egress-group egg1
quit

ingress-group ingg1
  route-map rm1
quit
```

Listener Policies

A listener policy is a set of commands that specifies the actions to be performed on packets as packets leave the 9920 device. Each listener policy can exist as multiple instances, which are differentiated by a user-specified, unique sequence ID. Listener policies are associated with other settings that are also specified as policies.

Configure a Listener Policy

Follow this procedure to map an ACL to an ingress policy and an egress and define actions for a matching ACL.

About This Task

Each listener policy can exist as multiple instances, which are differentiated by a user-specified, unique sequence ID. A listener policy maps an ACL of each type to an egress and defines the actions for the matching ACL.

> Note
>
> In the examples that follow the configuration commands for this task, **bold font** is used to show the relationship between the items you configure and then bind. The example parameters and optional settings are only a partial set of those available. See the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information.

Procedure

1. Run the **configure terminal** command to access Config mode.

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Configure an access list and the actions for this ACL.

   ip access-list *acl-name*

3. Create the listener policy, match the ACL, and include any action subcommands for the policy.

   listener-policy *name seq*

   ```
   device(config)# listener-policy lp-2
   device(config-listener-policy)# match ip address acl5-ipv4

   device(config-listener-policy)# strip-brtag
   device(config-listener-policy)# vlan vl-4085
   device(config-listener-policy)# description “ABCD”
   ```

4. Configure an egress policy, and bind the listener policy, specifying any additional egress actions.

   egress *name*

   ```
   device(config-egress)# egress e2
   device(config-egress)# set listener-policy lp-2
   device(config-egress)# precedence 1 interface ethernet 1/14
   ```

5. Configure an egress group and associate it with the egress policy.

   egress-group *group-name*

   ```
   device(config-egress-group)# egress-group eg_1
   device(config-egress-group)# description egress-group_1
   device(config-egress-group)# set egress e2
   ```

6. Configure the route map, and set any other parameters, such as forwarding actions, match ip access list, and the egress-group.

   route-map *map-name seq*

   ```
   device(config-route-map)# route-map R1 10
   device(config-route-map)# forward-action permit
   device(config-route-map)# match ip access-list test_1
   device(config-route-map)# set egress-group eg_1
   ```

7. Configure the interface port and channel for egress traffic.

   interface *slot / port-number*

   Note
   In the following example, traffic is leaving on slot/port-number 2/14.

   ```
   interface ethernet 1/14
   speed 100000
   description To_Tool
   no shutdown
   ```

8. Configure the interface port and channel for ingress traffic.

   interface *slot / port-number*

   Note
   In the following example, traffic is coming in on slot/port-number 2/3.

   ```
   interface ethernet 1/3
   description From_TAP
   ingress-group TAP_TRAFFIC
   no shutdown
   ```

9. Configure an ingress group and associate a route map.

   ingress-group *group-name*

   ```
   ingress-group TAP_TRAFFIC
   set route-map R1 10
   ```

Egress

An *egress* defines the interface or range of interfaces to be used for tool ports. If you have not configured an egress yet, see Network Traffic Monitoring Preparation on page 13.

Egress policies are associated with ACLs, route-maps, listener policies, and other settings to perform actions to be taken on packets at egress from those tool ports, such tag-stripping, header additions, and other actions that you define with subcommands within the policy.

It is important to note the following when configuring an egress policy:

- An egress policy can specify only one listener policy.
- An ACL bound to an egress policy can be modified.
- A listener policy bound to an egress can be modified.
- An ACL bound to a listener policy or route-map can be modified.

Configure an Egress Policy

Follow this procedure to prepare packets for actions to be performed when leaving the 9920.

Procedure

1. Run the **configure terminal** command to access Config mode.

   configure terminal

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Configure an ACL of type IPv4, IPv6, or MAC and any actions.

   ip access-list *acl-name*

   The specified ACL and configured actions will be bound to a listener policy.

   ```
   device(config)# ip access-list acl5-ipv4
   device(config-ip-acl)# seq 15 permit ip any any count
   ```

3. Create the listener policy, including any action subcommands for the policy.

   Note
   A listener policy supports only one of each of each ACL type: IP, IPv6, MAC .

   listener-policy *name seq*

   ```
   device(config)# listener-policy lp-2 220
   device(config-listener-policy)# match ip access-list acl5-ipv4
   device(config-listener-policy)# strip vn-tag
   device(config-listener-policy)# description “ABCD”
   ```

4. Configure an egress policy, and bind the listener policy, specifying any additional egress actions.

   Note
   An egress can be associated with only one listener policy.

   egress *name*

   ```
   device(config-egress)# egress e2
   device(config-egress)# set listener-policy lp-2
   device(config-egress)# description DirectTool
   device(config-egress)# set encap encap-1
   device(config-egress)# precedence 1 interface ethernet 2/14
   ```

5. Configure the interface port and channel for egress traffic.

   `interface` *`slot / port-number`*

   Note
   In the following example, traffic is leaving on slot/port-number 2/14.

   ```
   interface ethernet 2/14
   speed 100000
   description To_Tool
   no shutdown
   ```

Route-Map

A route-map consists of ACLs and set of directives that define the egress interfaces for forwarding the flow.

A route-map consists of a sequence of instances, the equivalent of rows in a table. The device evaluates an incoming packet according to route-map instances in ascending numerical order. The incoming packet is first compared against instance 1, then against instance 2, and so on. When a match is found, the device stops evaluating the incoming packet.

Route-maps can contain `match` clauses and `set` statements. Each route map contains a `forward-action permit` or `forward-action deny` statement that modifies the behavior of the route-map instance:

- If the route-map contains a `forward-action permit` statement, the access lists present in the match statement are used to match against incoming packets, and matched packets are forwarded according to the access lists. If the packets do not match the access lists in the match statement, the packets are passed to the next instance in the same route-map. If the packet does not match any of the access lists in the route-map, it is dropped.
- If the route-map instance contains a `deny` statement, the matching route-map instance is skipped. The route-map instance does not program the access lists in the match statements, and packets that match, skip that route-map instance.
- If an incoming packet does not match any access lists in the match statements of the route-map, the packet is dropped. This is the default action. To change the default action, configure the last `match` statement in the last instance of the route-map to a MAC access list with a clause of `permit any any`.
- If there is no `match` statement, route-map instance is skipped like the `forward-action` is set to `deny`.

If the route-map contains `set` statements, packets that are permitted by the route-map `match` statements are forwarded according to the `set` statements.

Map ACLs with Route-Map

Procedure

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Create ACL and define the rules.

   ```
   device(config)# ip access-list name
   ```

3. Create a route-map.

   ```
   device (config)# route-map name sequence_number
   ```

4. (Optional) Delete the route-map instance.

   ```
   device (config) # no route-map name
   ```

5. Map ACL to the route-map.

   ```
   device (config-route-map-name) # match [ mac | ip | ipv6 ] access-list name
   ```

6. (Optional) Remove ACL mapping from the route-map.

   ```
   device (config-route-map-name) # no match [ mac | ip | ipv6 ] access-list name
   ```

7. Configure the route-map to be matched or skipped.

   ```
   device (config-route-map-name) # forward-action [ permit | deny ]
   ```

8. Configure the egress-group.

   ```
   device (config-route-map-name) # set egress-group name
   ```

Create Egress Group

Procedure

1. Create an egress and add (remove) port interface to the egress.

   ```
   device (config)# egress name
   device (config)# [no] precedence 10 interface ethernet 0/10
   device (config)# [no] set encap encap100
   device (config)# [no] set listener-policy lp100
   ```

2. (Optional) Delete an egress port.

   ```
   device (config)# no egress name
   ```

3. Group the egress ports associated with the egress group.

   ```
   device (config)# egress-group name
   device (config)# set egress egress-one
   device (config)# set egress egress-two
   ```

Example

Adding second interface to an egress is not supported.

```
device (config)# egress eg-test1
device (config)# [no] precedence 10 interface ethernet 1/4
device (config)# [no] precedence 10 interface ethernet 1/4

Error: precedence already exists ethernet 1/4 egress eg-test1.
```

Ingress

You configure ingress policies—which are associated with ACLs, route-maps, listener policies, and other settings—to perform actions to be taken on packets at ingress.

It is important to note the following when configuring an ingress policy:

- An ingress policy can specify only one route-map.
- A route-map bound to the ingress-group can be modified
- An ACL bound to a listener policy can be modified.
- An ACL bound to a listener-policy or route-map can be modified.

Ingress

You configure an ingress policy to perform actions that affect packet-flow.

An ingress policy is used to specify the actions to be performed at ingress on packets received by the 9920 device. An ingress policy specifies actions for packet flow.

You can create multiple listener-policy stanzas, but they do not have to be used in unique 1:1 configurations, as demonstrated in the following example to manage different traffic types: two listener policies (lp-1 and lp-2) are used to manage three traffic types.

- Listener-policy `lp-1` has two entries:
  - One manages 802.1BR traffic.
  - One manages VN-TAG traffic.

Table 4: Example: Listener-policy use for traffic types

Example 1	Example 2	Example 3

`listener-policy lp-1 10`
`match ip access-list`
`v4_someacl1`
`strip br-tag`
	`listener-policy lp-1 20`
	`match ip access-list`
	`v4_someacl2`
	`strip vn-tag`
		`listener-policy lp-2 10`
		`match ip access-list`
		`v4_someacl1`
		`strip vn-tag`

Configure an Ingress Policy

Follow this procedure to configure an ingress policy for actions to be performed on packets at ingress.

About This Task

At a minimum, you must configure an ingress policy with a route-map with a match statement and a valid ACL.

Note

In the examples that follow the configuration commands for this task, bold font is used to show the relationship between the items you configure and then bind. The example parameters and optional settings are only a partial set of those available. See the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information.

Procedure

1. Run the **`configure terminal`** command to access Config mode.

   `configure terminal`

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Configure an ACL of type IPv4, IPv6, or MAC and any actions.

   ip access-list *acl-name*

   The specified ACL and configured actions will be bound to a route map.

   ```
   device(config)# ipv6 access-list acl2-ipv6
   device(config-ip-acl)# seq 10 permit ipv6 any any count
   ```

3. Configure the route map match ipv6 access list and set any other parameters, such as forwarding actions and the egress-group.

   route-map *map-name seq*

   ```
   device(config-route-map)# route-map R1 10
   device(config-route-map)# match ipv6 access-list acl2-ipv6
   device(config-route-map)# set egress-group eg_1
   device(config-route-map)# forward-action permit
   ```

4. Configure the interface port and channel for ingress traffic.

   interface *slot / port-number*

   Note
   In the following example, traffic is coming in on slot/port-number 2/3.

   ```
   interface ethernet 2/3
   description From_TAP
   set ingress-group TAP_TRAFFIC
   no shutdown
   ```

5. Add the route map to the ingress group.

   ingress-group *group-name*

   ```
   device(config)# ingress-group TAP_TRAFFIC
   device(config-ingress-group)# set route-map R1
   ```

View Ingress-Group Details

Procedure

1. View ingress-group configuration.

   ```
   # show ingress-group [ name | all ]
   ```

2. View ingress-group counters.

   ```
   # show counters ingress-group [ name | all ]
   ```

3. View running-config ingress-group information.

   ```
   # show running-config ingress-group
   ```

Replication

Extreme 9920 software replicates network traffic to multiple ports and port-channels.

Extreme 9920 software achieves replication using egress groups. Replication Service is responsible for forwarding the packets to more than one egress ports. Replication ensures that each monitoring tool has access to necessary packets. The egress groups contain more than one egress object during replication. A route-map or ingress policy always points to an egress group and this ensures that moving between unicast and multicast cases does not require any route-map object changes. The egress-group associated with the route-map can be modified to add or remove egress objects.

Extreme 9920 software uses replication block in the hardware to generate copies for each member. Each copy derives a unique internal identifier which is bound to its own egress-policy (listener-policy) and tunnel-encap properties. If the destination is LAG, load balancing is done based on the system LAG hash properties.

Extreme 9920 software supports the following replication functionality:

- Ingress replication of Layer 3 (IPv4 and IPv6) packets
- Replication of VNTAG, VXLAN, and 802.1BR headers on the ingress ports to an egress port or port group.
- Replication of packets to interfaces without changing information in the packet

For information about replication scale limitations, see *Extreme 9920 Software Scale and Standards Matrix, 21.1.0.0*.

Configure Replication

Procedure

1. Enter global configuration mode and create an egress port.

```
device (config)# egress name
```

2. Associate egress ports or LAG with the egress.

```
device(config-egress)# egress name
device(config-egress)# precedence number interface port-channel number
```

3. Set egress group within the route-map.

```
device(conf)# route-map name permit sequence_number
device(conf-route-map)# set egress-group name
```

4. Add an egress object to the egress-group, pointing to a LAG.

```
device (conf-egress-group)# set egress egress-one
```

   Traffic is forwarded to the LAG and load balanced within the LAG.

5. Add or remove LAG members as required for load balancing.

```
device# configure terminal
device(config)# interface ethernet 1/1
device(config-if)# channel-group 1 mode on
device(config-if)# no channel-group 1
```

6. Add another egress object to the egress-group.

```
device (config)# egress-group name
device(conf-egress-group)# set egress egress-two
```

   Traffic is replicated and copies are sent to both egress objects.

7. Create the required listener policy.

```
device (config)# listener-policy { name sequence-id }
```

8. Add the required listener policy.

```
device (conf-egress-group)# set listener-policy name
```

   The listener policy is applied based on the egress object.

9. Configure the `encap type`.

   ```
   encap-type gre
   ```

10. Add a new egress object with single port to the egress-group.

    ```
    device (config)# egress-group name
    device (conf-egress-group)# egress egress-three
    ```

    This egress object receives plain frames unmodified.

Example

egress toolA

```
# conf
(conf)# egress egressA
(conf-egress)# precedence 10 interface ethernet 0/10
(conf-egress)# set encap encap-100
(conf-egress)# set listener-policy lp-100
```

egress toolB

```
# conf
(conf)# egress egressB
(conf-egress)# precedence 10 interface ethernet 0/20
(conf-egress)# set encap encap-200
(conf-egress)# set listener-policy lp-100
```

egress toolC

```
# conf
(conf)#egress egressC
(conf-egress)# precedence 10 interface ethernet 0/30
(conf-egress)# set encap encap-100
(conf-egress)# set listener-policy lp-200
```

egress-group (replication)

```
# conf
(conf)# egress-group groupABC
(conf-egress-group)# egress egressA
(conf-egress-group)# egress egressB
(conf-egress-group)# egress egressC
```

route-map

```
(conf)# route-map rmap1 permit 10
(conf-route-map)# match ipv4-acl ipv4_acl1
(conf-route-map)# set egress-group groupABC
```

ingress-group

```
device(config)#ingress-group group-1
device(config-group-1)# set route-map rmap1
```

show running-config

```
device# show running-config egress-group
egress-group groupABC
  egress egressA
  egress egressB
  egress egressC
```

show egress-group

```
device# show egress-group groupABC
    Name : groupABC
     Description : -
          egress : egressA
          egress : egressB
          egress : egressC
```

Header Modification

Tagging and encapsulation techniques are an established part of networking, but some tagging protocols can create blind spots or unintended packet loss when third-party applications are not designed to interpret some or all of them. Stripping encapsulation header tags removes the interpreting burden and prepares them for downstream tools.

Header-Modification Overview

You can configure settings to prepare and optimize packet headers for forwarding and processing by analysis tools.

The NPB application provides the following header-modification functionality:

- Tag stripping
- Header-tag addition

(For more information about decapsulation, see Tunnel Termination.)

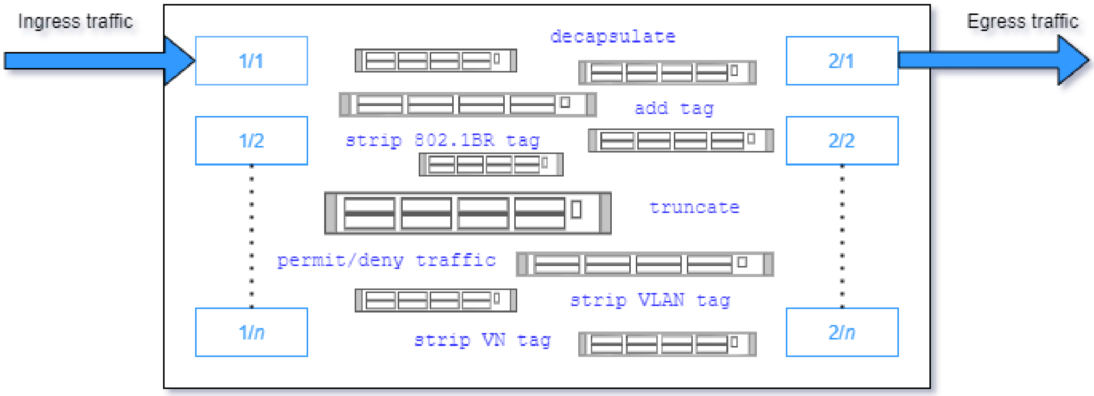


Figure 1: Header-modification and other packet actions available between packet ingress and egress

Header-Modification Flow

Header modification actions are performed as specified at ingress and at egress.

Based on configuration settings in the route map or the access list (ACL), when a packet is received by the application, it is evaluated against those settings and actions are performed. Ingress policy actions are applied before those associated with an egress policy.

Header-modification actions include tag stripping, tag addition, and decapsulation. You specify header-modification actions in a listener policy, which is bound to an egress policy and refers to ACLs and packet-modification settings.

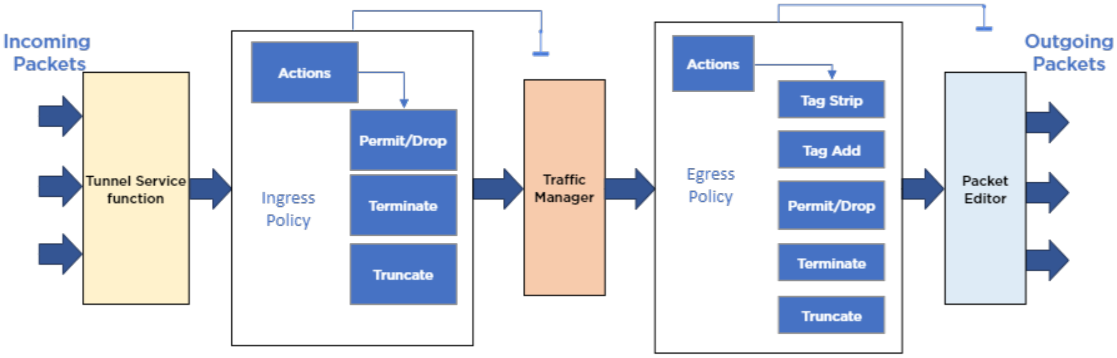


Figure 2: Header and packet modification flow

Tag-Stripping Overview

Understand how policies are used to strip header tags.

Stripping encapsulation header tags prepares them for downstream analytic and other monitoring tools. Header-tag stripping at egress is applied on only the first L2 header of the received packet.

You configure a listener policy to define the actions to be performed on packet header tags, which you then bind to an egress policy. A listener policy supports the following header-modification and traffic actions.

- 802.1BR
- VN-TAG
- VLAN

You can create multiple listener-policy stanzas, but they do not have to be used in unique 1:1 configurations, as demonstrated in the following example to manage different traffic types. In the following example, two listener policies (cow and smurf) are used to manage three traffic types.

- Listener-policy `cow` has two entries:
  - One manages 802.1BR traffic.
  - One manages VN-TAG traffic.

Table 5: Example: Listener-policy use for traffic types

Example 1	Example 2	Example 3

`listener-policy lp-1 10`
`match ip access-list`
`v4_someacl1`
`strip br-tag`
	`listener-policy lp-1 20`
	`match ip access-list`
	`v4_someacl2`
	`strip vn-tag`
		`listener-policy lp-2 10`
		`match ip access-list`
		`v4_someacl1`
		`strip vn-tag`

The following table lists the minimum requirements that must be configured for each policy type.

Table 6: Minimum policy configuration requirements

Ingress policy (packet flow)	Egress policy (packet prep for analysis tools)
Match ACL	Match ACL
Route-map with a valid match statement referencing a created ACL.	A designated interface or multiple interfaces.

It is important to note the following when configuring an ingress or egress policy:

- An ingress policy can specify only one route-map policy.
- An egress policy can specify only one listener policy.
- A route map or listener policy can support multiple match-ACLs.
- An ACL bound to the listener policy can be modified.
- A listener policy bound to the egress policy can be modified.
- A route-map bound to the ingress-group can be modified.

802.1BR Tag-Stripping

Learn the high-level process and how a packet appears before and after 802.1BR tag stripping.

Stripping the 802.1BR header tag prepares the packet for forwarding to analysis tools. The 802.1BR header tag is 8 bytes long and positioned after MAC-SA. This header can be followed by an S-Tag, a C-Tag, or both. The following figure shows the structure and position of 802.1BR header in a frame and its appearance after configuring a listener policy to strip 802.1BR headers.

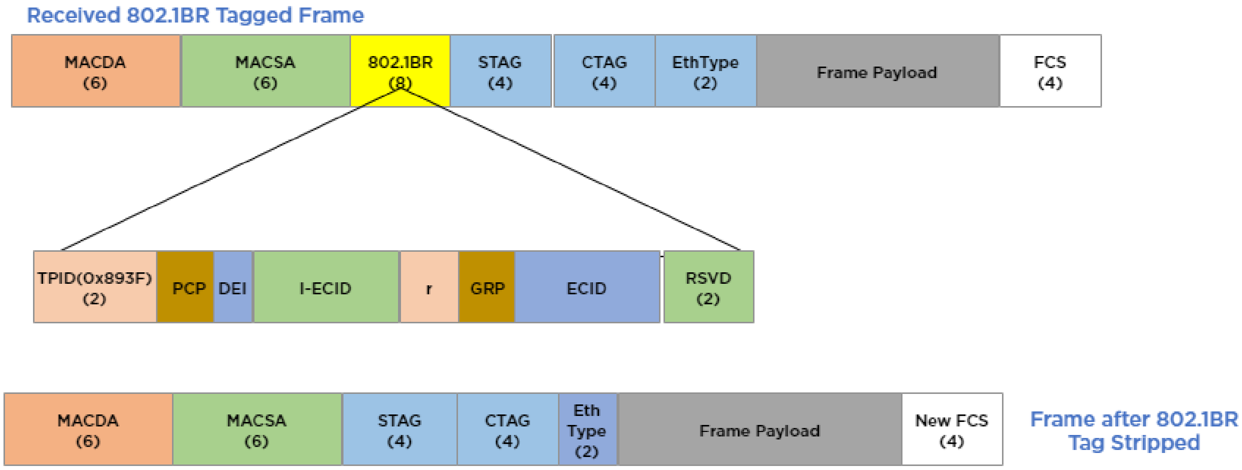


Figure 3: 802.1BR header before and after tag stripping

Configure 802.1BR Tag-Stripping

Follow this procedure to strip 802.1BR tags to support the encapsulation type expected by your traffic-analysis tools.

About This Task

- The vn-tag cannot be enabled if the br-tag is already enabled in the same listener policy.
- If a tunneled frame has an 802.1BR tag in the outer L2 header, VXLAN, NVGRE, or GTP header-stripping also deletes the 802.1BR tag.

Note
In the examples that follow the configuration commands for this task, bold font is used to show the relationship between the items you configure and then bind. The example parameters and optional settings are only a partial set of those available. See the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information.

Procedure

1. Run the **configure terminal** command to access Config mode.

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Configure an ACL of type IPv4, IPv6, or MAC and any actions.

   ip access-listip access-list *acl-name*

   The specified ACL and configured actions will be bound to a listener policy.

   ```
   device(config)# ip access-list acl5-ipv4
   device(config-ip-acl)# seq 10 permit ip any any count
   ```

3. Create the listener policy, including any action subcommands for the policy.

   Note
   A listener policy supports only one of each of each ACL type: IP, IPv6, MAC .

   listener-policy *name seq*

   ```
   device(config)# listener-policy lp-2 24
   device(config-listener-policy)# match ip access-list acl5-ipv4
   device(config-listener-policy)# strip br-tag
   device(config-listener-policy)# description "Strips 802.1BR tags"
   ```

4. Configure an egress policy, and bind the listener policy, specifying any additional egress actions.

   Note
   An egress can be associated with only one listener policy.

   egress *name*

   ```
   device(config)# egress e2
   device(config-egress)# set listener-policy lp-2
   device(config-egress)# description DirectTool
   device(config-egress)# set encap encap-1
   device(config-egress)# precedence 1 interface ethernet 1/14
   ```

5. Configure an egress group and associate it with the egress policy.

   egress-group *group-name*

   ```
   device(config)# egress-group eg_1
   device(config-egress-group)# description e-group_1
   device(config-egress-group)# set egress e2
   ```

6. Configure the route map and set any other parameters, such as forwarding actions, match ip access list, and the egress-group.

   Note
   A route-map policy supports only one match-ACL per layer.

   route-map *map-name seq*

   ```
   device(config)# route-map R1 10
   device(config-route-map)# match ip access-list acl5-ipv4
   device(config-route-map)# set egress-group eg_1
   device(config-route-map)# forward-action permit
   ```

7. Configure an ingress group and associate a route map.

   Note
   An ingress group can be associated with only one route map.

   ingress-group *group-name*

   ```
   device(config)# ingress-group TAP_TRAFFIC
   device(config-ingress-group)# set route-map R1
   ```

8. Configure the interface port and channel for ingress traffic.

   interface *slot / port-number*

   Note
   In the following example, traffic is coming in on slot/port-number 2/3.

   ```
   interface ethernet 1/3
   description From_TAP
   set ingress-group TAP_TRAFFIC
   no shutdown
   ```

9. Configure the interface port and channel for egress traffic.

   interface *slot / port-number*

   Note
   In the following example, traffic is leaving on slot/port-number 2/14.

   ```
   interface ethernet 2/14
   speed 100000
   description To_Tool
   no shutdown
   ```

VLAN Header Tag Stripping

Learn the high-level process and how a packet appears before and after VLAN tag stripping.

The VLAN tag is the first tag after MAC-SA and is 4 bytes long. The following figure shows the structure and position of the VLAN header in a frame and its appearance after configuring a policy to strip VLAN headers.

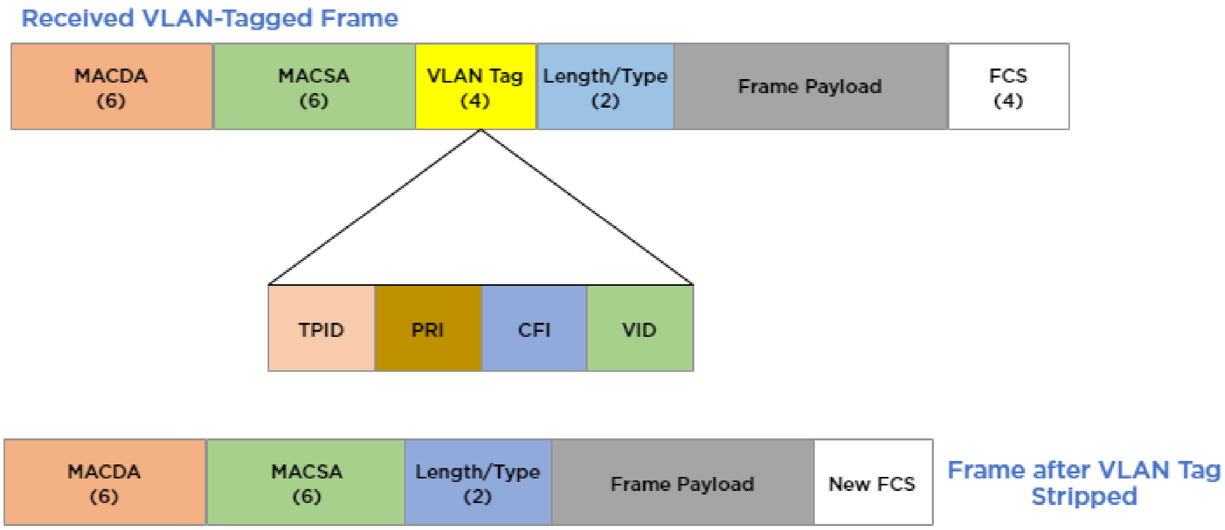


Figure 4: VLAN tag before and after tag stripping

Configure VLAN Tag Stripping

Follow this procedure to strip VLAN tags to support the encapsulation type expected by your traffic-analysis tools.

About This Task

The VLAN tag is not inside the packet and must be stripped at the ingress device before the encapsulated packet is sent into the network. You can configure VLAN tag stripping so it is performed at ingress or egress.

Note
In the examples that follow the configuration commands for this task, bold font is used to show the relationship between the items you configure and then bind. The example parameters and optional settings are only a partial set of those available. For more information, see the *Extreme 9920 Software Command Reference, 21.1.0.0*.

Procedure

1. Run the **configure terminal** command to access Config mode.

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Configure an ACL of type IPv4, IPv6, or MAC and any actions.

   ip access-list *acl-name*

   The specified ACL and configured actions will be bound to a listener policy.

   ```
   device(config)# ip access-list acl3-ipv4
   device(config-ip-acl)# seq 12 permit ip any any count
   ```

3. Create the listener policy, including any action subcommands for the policy.

   Note
   A listener policy supports only one of each of each ACL type: IP, IPv6, MAC .

   listener-policy *name seq*

   ```
   device(config)# listener-policy lp-10 5
   device(config-listener-policy)# match ip access-list acl3-ipv4
   device(config-listener-policy)# strip vlan-tag
   device(config-listener-policy)# description "ipv4 listener policy, strip vlan tag"
   ```

4. Configure an egress policy and bind the listener policy, specifying any additional egress actions.

   Note
   An egress can be associated with only one listener policy.

   egress *name*

   ```
   device(config-egress)# egress e3
   device(config-egress)# set listener-policy lp-3
   device(config-egress)# description DirectTool
   device(config-egress)# precedence 1 interface ethernet 1/14
   ```

5. Configure an egress group and associate it with the egress policy.

   egress-group *group-name*

   ```
   device(config)# egress-group eg_5
   device(config-egress-group)# description e-group_5
   device(config-egress-group)# set egress e3
   ```

6. Configure the route map and set any other parameters, such as forwarding actions, match ip access list, and the egress-group.

   Note
   A route-map policy supports only one match-ACL per layer.

   route-map *map-name seq*

   ```
   device(config)# route-map R2 10
   device(config-route-map)# match ip access-list acl3-ipv4
   device(config-route-map)# set egress-group eg_5
   device(config-route-map)# forward-action permit
   ```

7. Configure an ingress group and associate a route map.

   Note
   An ingress group can be associated with only one route map.

   ingress-group *group-name*

   ```
   device(config)# ingress-group TAP_TRAFFIC
   device(config-ingress-group)# set route-map R2
   ```

8. Configure the interface port and channel for ingress traffic.

   interface *slot / port-number*

   Note
   In the following example, traffic is coming in on slot/port-number 2/3.

```
interface ethernet 1/3
description From_TAP
ingress-group TAP_TRAFFIC
no shutdown
```

9. Configure the interface port and channel for egress traffic.

   interface *slot / port-number*

   Note
   In the following example, traffic is leaving on slot/port-number 2/14.

```
interface ethernet 2/14
description To_Tool
no shutdown
```

VN-Tag Stripping

Learn the high-level process and how a packet appears before and after VN-tag stripping.

The VN-tag is the first tag after MAC-SA and is 6 bytes long, It can be followed by an S-tag, a C-tag, or both.The following figure shows the structure and position of VN-tag in a frame before and after VN-tag stripping.

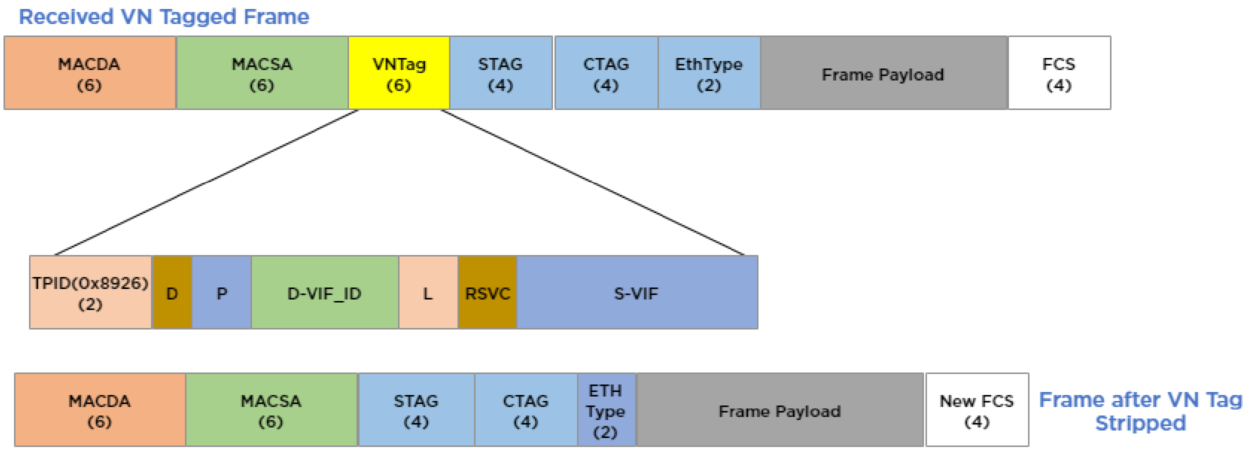


Figure 5: VN tag before and after stripping

Configure VN-Tag Stripping

Follow this procedure to strip VN tags to support the encapsulation type expected by your traffic-analysis tools.

About This Task

- The vn-tag cannot be enabled if the br-tag is already enabled in the same listener policy.

> Note
> In the examples that follow the configuration commands for this task, bold font is used to show the relationship between the items you configure and then bind. The example parameters and optional settings are only a partial set of those available. See the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information.

Procedure

1. Run the **configure terminal** command to access Config mode.

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Configure an ACL of type IPv4, IPv6, or MAC and any actions.

   ip access-list *acl-name*

   The specified ACL and configured actions will be bound to a listener policy.

   ```
   device(config)# ipv6 access-list acl2-ipv6
   device(config-ipv6-acl)# seq 9 permit ipv6 any any count
   ```

3. Create the listener policy, including any action subcommands for the policy.

   > Note
   > A listener policy supports only one of each of each ACL type: IP, IPv6, MAC .

   listener-policy *name seq*

   ```
   device(config)# listener-policy lp-11 43
   device(config-listener-policy)# match ipv6 access-list acl2-ipv6
   device(config-listener-policy)# strip vn-tag
   device(config-listener-policy)# description "LP for vn tag strippin ipv6"
   ```

4. Configure an egress policy, and bind the listener policy, specifying any additional egress actions.

   > Note
   > An egress can be associated with only one listener policy.

   egress *name*

   ```
   device(config)# egress e7
   device(config-egress)# set listener-policy lp-11 43
   device(config-egress)# precedence 1 interface ethernet 2/14
   ```

5. Configure an egress group and associate it with the egress policy.

   egress-group *group-name*

   ```
   device(config)# egress-group eg_9
   device(config-egress-group)# description e-group_9
   device(config-egress-group)# set egress e7
   ```

6. Configure the route map and set any other parameters, such as forwarding actions, match ip access list, and the egress-group.

   Note
   A route-map policy supports only one match-ACL per layer.

   route-map *map-name seq*

   ```
   device(config)# route-map R4 10
   device(config-route-map)# match ipv6 access-list acl2-ipv6
   device(config-route-map)# set egress-group eg_9
   device(config-route-map)# forward-action permit
   ```

7. Configure an ingress group and associate a route map.

   Note
   An ingress group can be associated with only one route map.

   ingress-group *group-name*

   ```
   device(config)# ingress-group TAP_TRAFFIC_2
   device(config-ingress-group)# set route-map R4
   ```

8. Configure the interface port and channel for ingress traffic.

   interface *slot / port-number*

   Note
   In the following example, traffic is coming in on slot/port-number 2/3.

   ```
   interface ethernet 1/3
   description From_TAP
   set ingress-group TAP_TRAFFIC_2
   no shutdown
   ```

9. Configure the interface port and channel for egress traffic.

   interface *slot / port-number*

   Note
   In the following example, traffic is leaving on slot/port-number 2/14.

   ```
   interface ethernet 2/14
   description To_Tool
   no shutdown
   ```

Tag Addition

Learn the high-level process and how a packet appears before and after adding a VLAN tag.

If configured, the VLAN tag is added to the stripped packet of both tagged and non-tagged packets in the first L2 header. The following figure shows tag addition for a non-tagged packet.

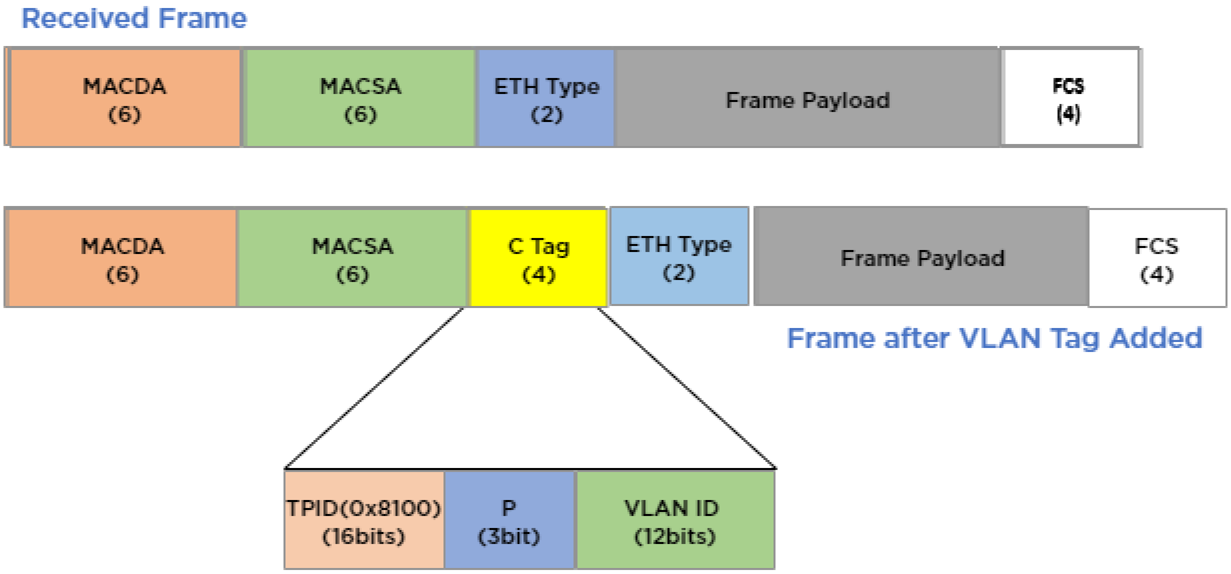


Figure 6: VLAN tag added, non-tagged frames

For a tagged frame, tag-addition is based on the existing tag. The operation is applied to the E-tag, VN-tag, or C-tag, as showin the following figure. Note the newly added C-tag.

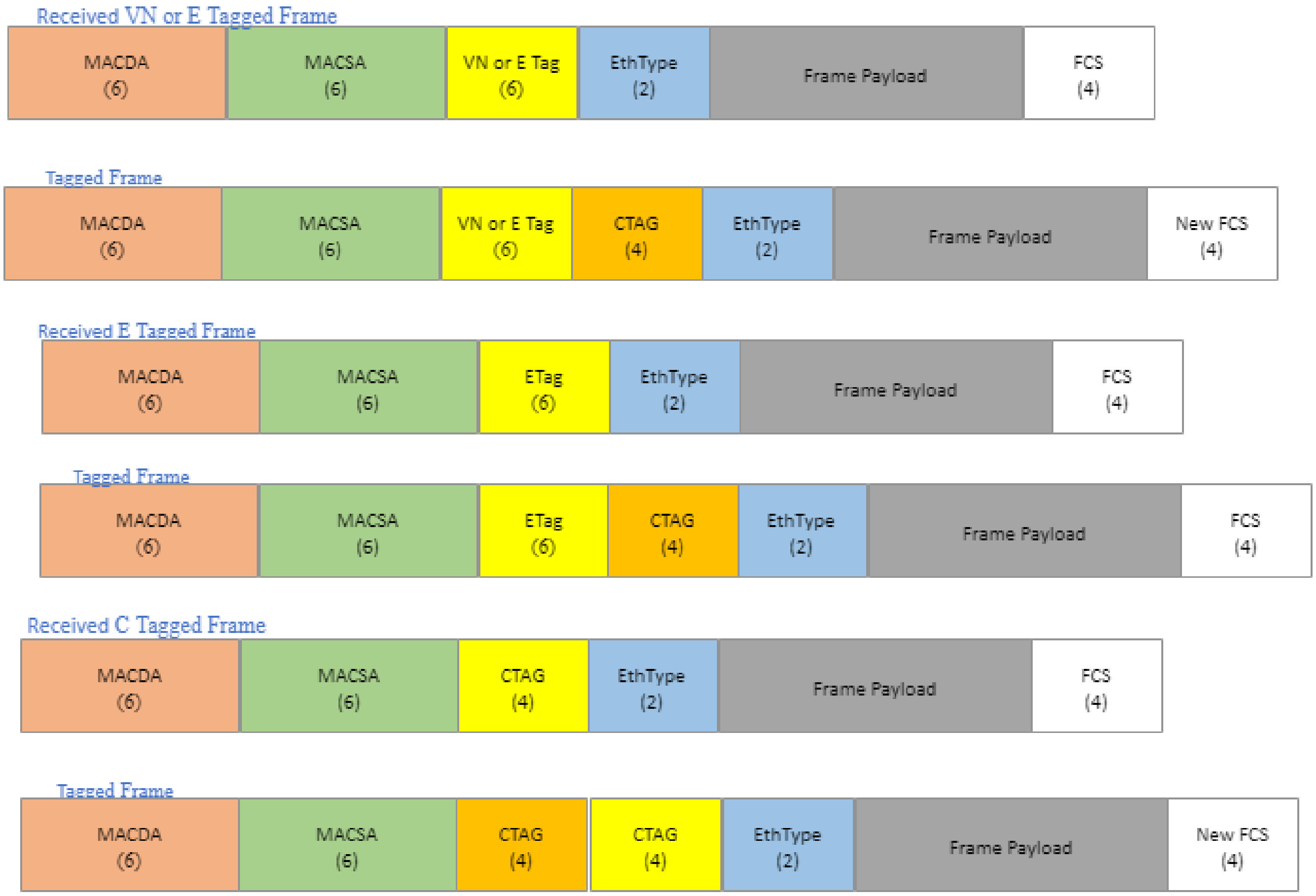


Figure 7: Tag addition, E-tag, VN-tag and C-tag frames

S-tag and double-C tag additions are no-operation (NOOP) additions, meaning it's a computer processor instruction. The following figures shows an example of tag addition forNOOP cases.

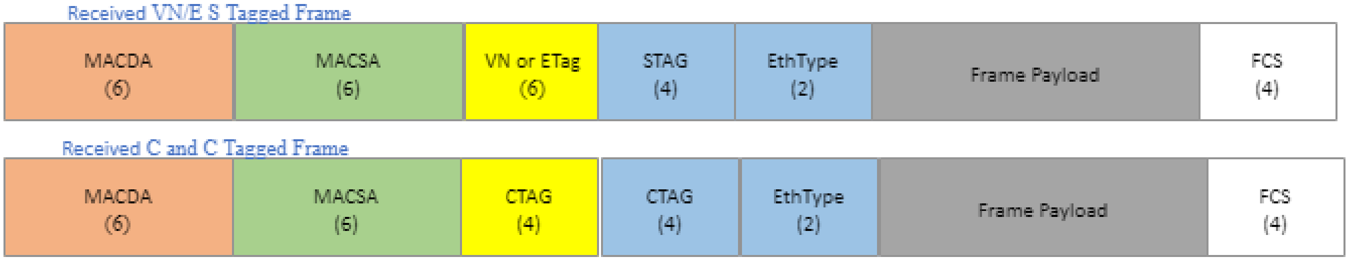


Figure 8: Tag addition, NOOP for tagged frames

Configure Tag Addition

Follow this procedure to add a VLAN tag to tagged or non-tagged packets.

About This Task

Tag addition is performed in the first L2 header.

Note
In the examples that follow the configuration commands for this task, bold font is used to show the relationship between the items you configure and then bind. The example parameters and optional settings are only a partial set of those available. See the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information.

Procedure

1. Run the **configure terminal** command to access Config mode.

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Configure an ACL of type IPv4, IPv6, or MAC and any actions.

   ip access-list *acl-name*

   The specified ACL and configured actions will be bound to a listener policy.

   ```
   device(config)# ip access-list acl5-ipv4
   device(config-ip-acl)# seq 10 permit ip any any count
   ```

3. Create the listener policy, including any action subcommands for the policy.

   Note
   A listener policy supports only one of each of each ACL type: IP, IPv6, MAC . The VLAN *id* tag value ranges from 1 to 4095.

   listener-policy *name seq*

   ```
   device(config)# listener-policy lp-2 32
   device(config-listener-policy)# match ip access-list acl5-ipv4
   device(config-listener-policy)# vlan id
   device(config-listener-policy)# description "ipv4 tag add lp"
   ```

4. Configure an egress policy, and bind the listener policy, specifying any additional egress actions.

   Note
   An egress can be associated with only one listener policy.

   egress *name*

   ```
   device(config)# egress e2
   device(config-egress)# set listener-policy lp-2 32
   device(config-egress)# description DirectTool
   device(config-egress)# precedence 1 interface ethernet 2/14
   ```

5. Configure an egress group and associate it with the egress policy.

   egress-group *group-name*

   ```
   device(config)# egress-group eg_1
   device(config-egress-group)# description e-group_1
   device(config-egress-group)# set egress e2
   ```

6. Configure the route map and set any other parameters, such as forwarding actions, match ip access list, and the egress-group.

   Note
   A route-map policy supports only one match-ACL per layer.

   route-map *map-name seq*

   ```
   device(config)# route-map R1 10
   device(config-route-map)# match ip access-list acl5-ipv4
   device(config-route-map)# set egress-group eg_1
   device(config-route-map)# forward-action permit
   ```

7. Configure an ingress group and associate a route map.

   Note
   An ingress group can be associated with only one route map.

   ingress-group *group-name*

   ```
   device(config)# ingress-group TAP_TRAFFIC
   device(config-ingress-group)# set route-map R1 10
   ```

8. Configure the interface port and channel for ingress traffic.

   interface *slot / port-number*

   Note
   In the following example, traffic is coming in on slot/port-number 2/3.

   ```
   interface ethernet 1/3
   description From_TAP
   set ingress-group TAP_TRAFFIC
   no shutdown
   ```

9. Configure the interface port and channel for egress traffic.

   interface *slot / port-number*

   Note
   In the following example, traffic is leaving on slot/port-number 2/14.

```
interface ethernet 2/14
speed 100000
description To_Tool
no shutdown
```

Packet Truncation and Termination

Some tools do not require all of the packet payload, so Packet Truncation gives you the ability to remove a portion of the packet length before forwarding at egress.

Tunnel termination removes a tunnel encapsulation header. For more information, see Tunnel Termination.

Packet Truncation Flow

You truncate received packets to preprare them for analysis tool.

Packet truncation is performed at either ingress or egress before packets are forwarded to analysyis tools, depending on the policy configuration. When the incoming packet length is greater than the configured truncation value, the packet is truncated to the length specified in a policy. The following figure is a conceptual representation of before and after packet truncation.

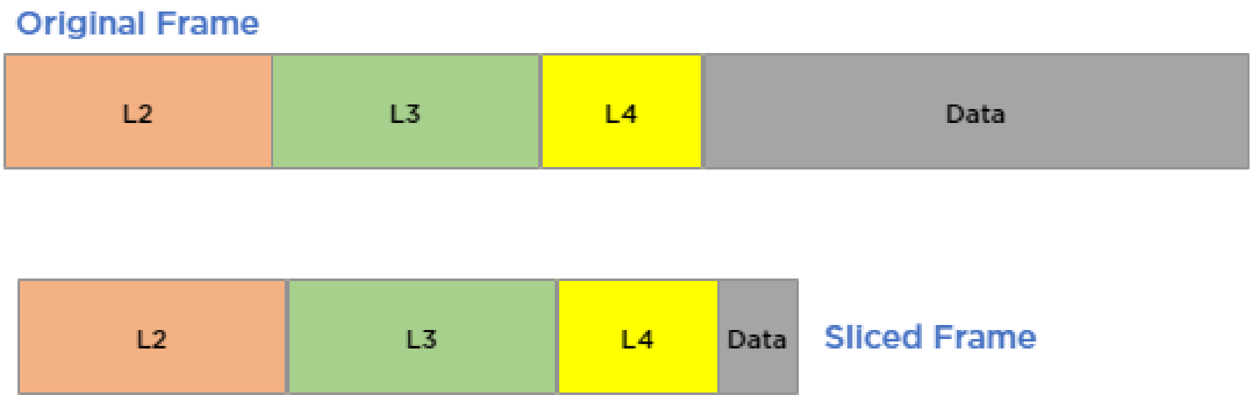


Figure 9: Packet before and after truncation

Because ingress policy actions are applied before those specified in the egress policy, the incoming packet may be terminated or truncated at ingress. Egress policy match criteria are applied to received tunneled or non-tunneled packets. The following lists the order of operations on packets upon ingress to the application:

1. Tag stripping
2. Tag addition
3. Forward or drop packets

4. Tunnel termination
5. Truncation

Configure Packet Truncation

Follow this procedure to truncate a packet to conform to the length required by your analysis tools.

Before You Begin

About This Task

You configure packet truncation in a listener policy. You can include other settings in addition to packet truncation. Minimum truncated packet length is 64 bytes.

Note
In the examples that follow the configuration commands for this task, bold font is used to show the relationship between the items you configure and then bind. The example parameters and optional settings are only a partial set of those available. See the *Extreme 9920 Software Command Reference, 21.1.0.0* for more information.

Procedure

1. Run the **configure terminal** command to access Config mode.

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Configure an ACL of type IPv4, IPv6, or MAC and any actions.

   ip access-list *acl-name*

   The specified ACL and configured actions will be bound to a listener policy.

   ```
   device(config)# ip access-list acl3-ipv4
   device(config-ip-acl)# seq 12 permit ip any any count
   ```

3. Create the listener policy, including the truncate subcommand with any other action subcommands for the policy.

   Note
   A listener policy supports only one of each of each ACL type: IP, IPv6, MAC .

   listener-policy *name seq*

   ```
   device(config)# listener-policy lp-3 18
   device(config-listener-policy)# match ip access-list acl3-ipv4
   device(config-listener-policy)# truncate 512
   device(config-listener-policy)# description "truncate ip packets lp"
   ```

4. Configure an egress policy and bind the listener policy, specifying any additional egress actions.

   Note
   An egress can be associated with only one listener policy.

   egress *name*

   ```
   device(config)# egress e3
   device(config-egress)# set listener-policy lp-3 18
   device(config-egress)# description DirectTool
   device(config-egress)# precedence 1 interface ethernet 2/14
   ```

5. Configure an egress group and associate it with the egress policy.

   egress-group *group-name*

   ```
   device(config)# egress-group eg_5
   device(config-egress-group)# description e-group_5
   device(config-egress-group)# set egress e3
   ```

6. Configure the route map and set any other parameters, such as forwarding actions, match ip access list, and the egress-group.

   Note
   A route-map policy supports only one match-ACL per layer.

   route-map *map-name seq*

   ```
   device(config)# route-map R2 10
   device(config-route-map)# match ip access-list acl3-ipv4
   device(config-route-map)# set egress-group eg_5
   device(config-route-map)# forward-action permit
   ```

7. Configure an ingress group and associate a route map.

   Note
   An ingress group can be associated with only one route map.

   ingress-group *group-name*

   ```
   device(config)# ingress-group TAP_TRAFFIC
   device(config-ingress-group)# set route-map R2
   ```

8. Configure the interface port and channel for ingress traffic.

   interface *slot / port-number*

   Note
   In the following example, traffic is coming in on slot/port-number 2/3.

   ```
   interface ethernet 2/3
   description From_TAP
   set ingress-group TAP_TRAFFIC
   no shutdown
   ```

9. Configure the interface port and channel for egress traffic.

   interface *slot / port-number*

   Note
   In the following example, traffic is leaving on slot/port-number 2/14.

   ```
   interface ethernet 2/14
   speed 40000
   description To_Tool
   no shutdown
   ```

Tunnel Termination Flow

Tunnel termination can be configured for received L2 or L3 packets.

You can configure settings to apply tunnel termination to received packets, either tunneled (both L2 and L3) or non-tunneled. Tunnel termination is performed at either ingress or egress before packets are forwarded to analysis tools, depending on the policy configuration. The outer tunnel of L2 tunneled packets is removed and the current position is shifted to the start of the inner L2 header, as shown in the following figure.

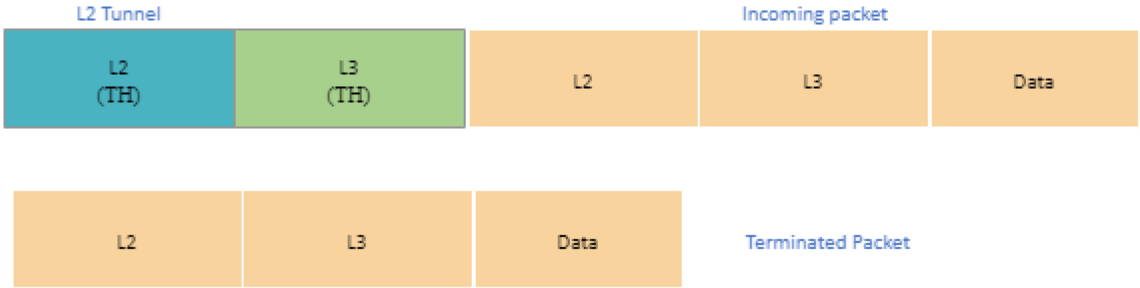


Figure 10: L2 tunnel termination

Because L3 tunneled-packet inner headers do not have the L2 header, the following occurs (shown in figure):

- The L2 header is retrieved from the L2 outer header
- The L3 outer header is stripped.

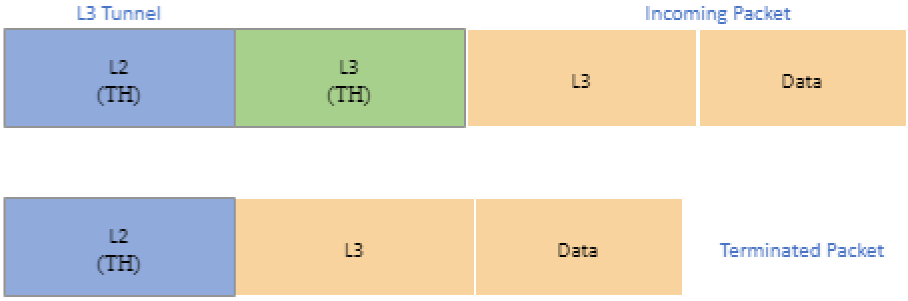


Figure 11: L3 tunnel termination

For more information about decapsulating packets, see Tunnel Termination on page 56.

Forward Incoming Traffic

About This Task

Perform the following procedure to forward incoming traffic using MAC, IP, or IPv6 access list configuration.

Procedure

1. Configure the ingress group with the required ingress ports.
2. Configure L2 (MAC) or L3 ACL (IP/IPv6) in the access list with Permit option.
3. Set L2 or L3 ACL in the route-map.
4. Configure the port channel, egress ports, or egress group based on redirect/forward to single egress port or multiple ports in route-map.
5. Apply route-map policy to the ingress group.

6. Send traffic flows to the DUT.
   - The traffic flows or packets are forwarded to egress port/egress group based on L2 or L3 ACL configured in the route-map.
   - Traffic flows are load-balanced between the mapped egress ports.
7. Verify the CLI statistics of ingress group and L2 or L3 ACL counters to determine the number of packets or flows received.
8. Verify the CLI statistics of egress port or egress group to determine the number of packets or flows forwarded.
9. Verify the CLI statistics of L2 or L3 ACL matches the number of packets or flows forwarded.
10. Verify that the packets matching the L2 or L3 ACL are getting forwarded in egress port/egress group by validating or capturing the wired PCAP collected in the analytical tools.
11. Verify the statistics by enabling the logs of forwarding agents.

Drop Incoming Traffic

About This Task

Perform the following procedure to drop incoming traffic using MAC, IP, or IPv6 access list configuration.

Procedure

1. Configure the ingress group with the required ingress ports.
2. Configure L2 (MAC) or L3 ACL (IP/IPv6) in the access list with Deny option.
3. Set L2 or L3 ACL in the route-map.
4. Apply route-map policy to the ingress group.
5. Send traffic flows to the DUT.

   The traffic flows or packets are dropped based on L2 or L3 ACL configured in the route-map.
6. Verify the CLI statistics of ingress group and L2 or L3 ACL counters to determine the number of packets or flows dropped.
7. Verify the CLI statistics of egress port or egress group.
8. Verify that the packets matching the L2 or L3 ACL are not getting forwarded in egress port by validating or capturing the wired PCAP collected in the analytical tools.

Filter GTP/SCTP/VXLAN Message Header

Procedure

1. Configure the ingress group with the required ingress ports.
2. Configure the required IP/IPv6 ACL settings to filter GTP, SCTP, or VXLAN message headers.

   Use the L4 port 132 to filter SCTP packets.
3. Set ACL filtering in the route-map.
4. Apply the route-map policy to the ingress group.

5. Send traffic flows to the DUT.
   - The configured GTP, SCTP, or VXLAN message header is dropped based on ACL configured in the route-map.
   - The non-filtered traffic is forwarded to the egress port/egress group.
6. Verify the CLI statistics of ingress group to determine the number of packets or flows received and dropped.
7. Verify the CLI statistics of egress port or egress group to determine the number of non-filtered packets or flows forwarded.
8. Verify the CLI statistics of UDA ACL matches the number of packets/flows dropped.
9. Verify that the packets matching the UDA ACL are not forwarded in egress groups and only non-filtered packets are received by validating or capturing the wired PCAP collected in the analytical tools.

Filter GTP Tunneled HTTP Messages

About This Task

Perform this procedure to filter or drop HTTPS traffic frames encapsulated in a version 1 GTP frame based on User Defined Attribute (UDA) ACL policy and allow non-filtered traffic to an EGRESS port or group.

Procedure

1. Configure the ingress group with the required ingress ports.
2. Configure the ACL for filtering GTP tunneled HTTPS messages.
3. Set ACL filtering in the route-map.
4. Apply the route-map policy to the ingress group.
5. Send GTP tunneled HTTPS/HTTP traffic flows to the DUT.
   - The HTTPS traffic tunneled in GTP is dropped based on ACL configured in the route-map.
   - The non-filtered HTTP traffic in GTP is forwarded to the egress port/egress group.
6. Verify the CLI statistics of ingress group to determine the number of packets or flows received and dropped.
7. Verify the CLI statistics of egress port or egress group to determine the number of non-filtered packets or flows forwarded.
8. Verify the CLI statistics of UDA ACL matches the number of GTP tunneled HTTPS packets/flows dropped.
9. Verify that the GTP tunneled HTTPS packets matching the UDA ACL are not getting forwarded in egress groups and GTP tunneled HTTP packets are received by validating or capturing the wired PCAP collected in the analytical tools.
10. Verify the statistics by enabling the logs of forwarding agents.

Onboard Packet Capture

Onboard Packet Capture tool allows quick capture of packets received or transmitted on a front panel port. The captured packets are stored in one or more Packet Capture (PCAP) files for debugging.

After capturing the configured number of packets, packet capturing process automatically stops for that particular interface. To save and retrieve packet capture in a PCAP file, packet capture must be stopped globally. Saving packet capture configurations in a config file is not required as packet capture is intended for live debugging for a short period of time only.

The packet capture CLI commands are allowed in the Exec mode. The **`show capture packet config`** command displays all packet capture configurations on Ethernet ports. The **`show capture packet interface`** command displays content of the PCAP file when packet capture is in-progress. For more information, see *Extreme 9920 Software Command Reference, 21.1.0.0*.

Note

- Packet filter based on L2 or L3 is not supported in NPB application.
- As packets sent to CPU are rate limited, capturing all Ethernet frames that are received or transmitted on the front panel port is not guaranteed.

PCAP File Management

The packets received from data-path are written to the active PCAP file, `pktcapture_running.pcapng`.

The active PCAP file is renamed and saved as `pktcapture_N.pcapng`, where N is 1-25.

- A maximum of 25 PCAP files with a file size of 100 MB each is supported. Packet capture automatically stops when 25 PCAP files are available. The existing PCAP files have to be removed to restart packet capture.
- The capture writes to the active PCAP file until file size reaches 100 MB. The PCAP file is then renamed and saved.
- If the capture is manually stopped, irrespective of the current file size, the active PCAP file is renamed and saved.

For information about onboard PCAP on EVM, see *Extreme Visibility Manager Administration and User Guide, 6.0.0*.

PCAP File Encoding

NPB application supports 10 simultaneous packet captures.

PCAP File Decoding

The **`show capture packet interface`** command shows the active PCAP file, `pktcapture_running.pcapng` in read-only mode. If the active PCAP file is not present, the latest inactive PCAP file information is displayed. For more information on Extreme 9920 software commands, see *Extreme 9920 Software Command Reference, 21.1.0.0*.

Note

Extreme 9920 software supports multiple CLI agents reading the active PCAP file simultaneously.

Configure Packet Capture

Procedure

1. Enable packet capture on an interface.

   ```
   device# capture packet interface ethernet IFNAME { direction [ both | rx | tx ]
   [packet-count 1-8000 ] }
   ```

2. Start packet capture.

   ```
   device# capture start
   ```

3. Stop packet capture.

   ```
   device# capture stop
   ```

Configure Ingress and Egress ACL Capture

Before You Begin

- The onboard PCAP configuration and ACL log option must be done only while debugging.
- All onboard PCAP configurations and ACL logging configuration must be unconfigured after completing debugging.

About This Task

You can use the `log` option to the Access Control Entry (ACE) to capture packet through ACL for IPv4, IPv6, and MAC. All packets forwarded by ACE are captured by onboard packet capture.

Procedure

1. Configure packet capture through ACL.

   ```
   ip access-list route-map-acl
       seq 10 permit ip any 2.2.2.2 255.255.255.255 log
   ip access-list listener-policy-acl
       seq 10 permit ip 1.1.1.1 255.255.255.255 any log
   ```

2. Attach the ACL to a route-map to make it an ingress ACL.

   ```
   route-map route_map_1 1
       forward-action permit
       match ip access-list route-map-acl
   ```

3. Attach the ACL to a listener policy to make it an egress ACL.

   ```
   listener-policy listener_policy_1 1
       forward-action permit
       match ip access-list listener-policy-acl
   ```

View Captured Packets

Procedure

1. View captured packets.

   ```
   show capture packet interface ethernet 1/1
   ```

2. View current capture configuration.

   ```
   show capture packet config
   ```

3. View metadata of all packet capture files.

   ```
   show capture packet pcapfile-info
   ```

Example

```
device# show capture packet config
All protocol RX capture is enabled on interface Eth 1/2
All protocol RX capture is enabled on interface Eth 1/3
All protocol RX capture is enabled on interface Eth 1/1
All protocol TX capture is enabled on interface Eth 1/1
```

Tunnel Termination

Tunnel termination classifies and decapsulates incoming packets. Tunnel termination can be done using route-map.

The packets are classified based on configured header parameters and then SAP IDs are assigned. Extreme 9920 software decapsulates packets based on parameters that you configure. There are two types of packet decapsulation:

- Non-transport tunnel termination
- Transport tunnel termination

Note
Only IPv4 tunnel termination is supported in the NPB application.

Both transport and non-transport tunnel terminations support IPv4 traffic. Based on the type of incoming traffic and configured tunnel termination parameters, the following actions are carried out:

- Termination
- Scope-shift
- Packet classification

The following configurations are required for successful tunnel termination:

- Encapsulation parameter configurations (L2 and L3 parameters for GRE header)
- Egress group configurations
    - Attaching egress object to egress group
- ACL configurations
- Ingress group configurations (for non-transport tunnel terminations)
- Transport tunnel configurations (for transport tunnel terminations)
- Interface configurations
- Route-map configurations

Non-Transport Tunnel Termination

Non-transport tunnel termination applies to packets that do not have destination MAC address within the chassis MAC address range and considered “not destined to us”.

Type of headers to be decapsulated can be configured using CLI or gNMI commands. Non-transport tunnel termination supports header decapsulation for VXLAN, IPGRE, nvGRE, GTPu, and IP-in-IP.

For VXLAN, GTPu, and nvGRE packets, a key or identifier can be configured for termination. For example, in VXLAN, a particular VNID can be configured to match for termination. If a key or identifier is not configured for traffic types, all packets are matched irrespective of the key or identifier they ingresses with.

The non-transport tunnel termination parameters can be configured through ingress-group CLI command. Tunnel termination takes the configured action on incoming GRE, GTP, IPinIP, NVGRE, and VXLAN packets:

- Terminate/Decapsulate: The encapsulation header is decapsulated and the remaining packets are processed in further blocks. This results in shifting the scope of the headers to inner headers automatically.

  Terminate or decapsulate can also be done using route-map or listener-policy.
- Scope-shift: There is no header decapsulation and the scope of the header is shifted to inner headers. This results in further blocks in the packet processing pipeline using inner headers of the packet.

  Scope-shift can also be done using route-map.
- No-operation: There is no decapsulation or scope shift, only packet classification is done. In ingress group, this results in further processing blocks processing all headers in the packet.

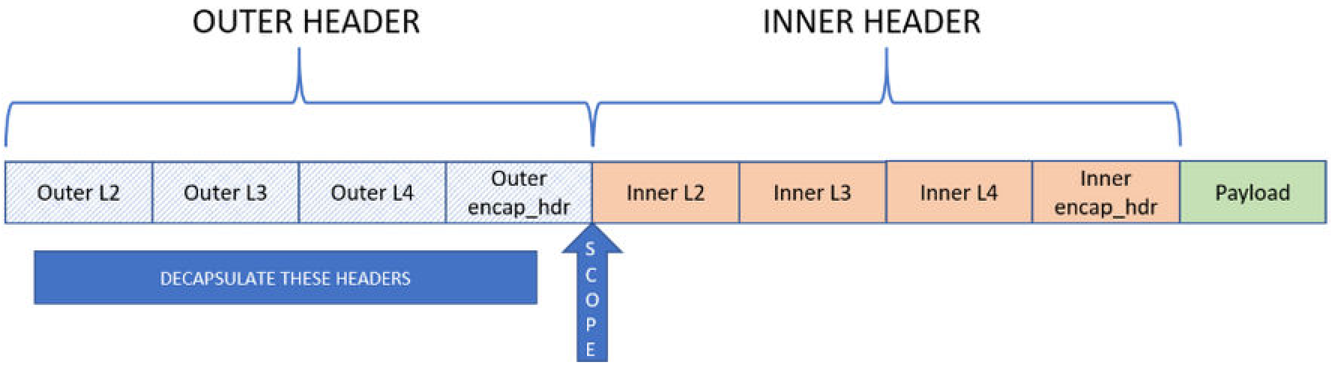


Figure 12: Non-transport tunnel termination in terminate mode

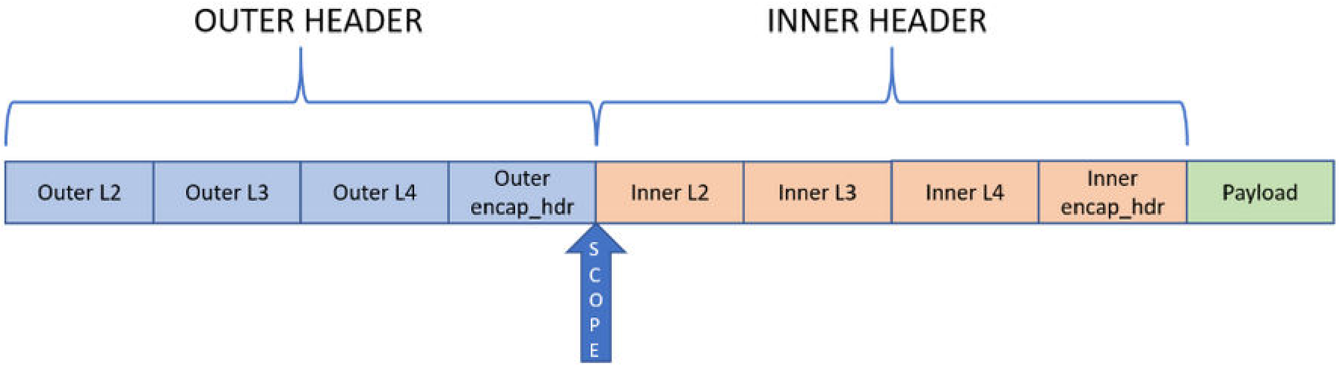


Figure 13: Non-transport tunnel termination in scope-shift mode

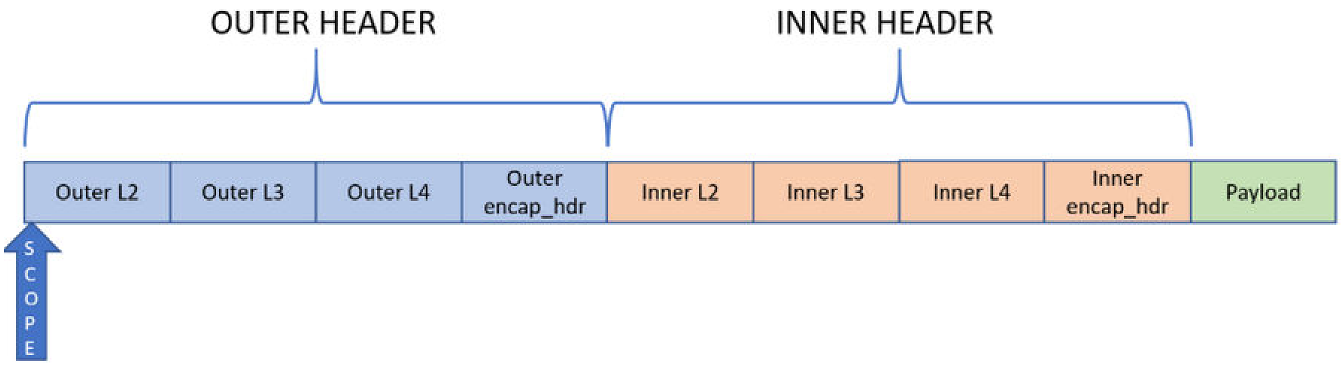


Figure 14: Non-transport tunnel termination in no-op mode

Transport Tunnel Termination

Transport tunnel termination applies to packets with destination MAC address within the chassis MAC address and considered “destined to us”.

The chassis MAC address is a non-configurable parameter and is configured in data path at boot-up time.

Transport tunnel termination supports decapsulation of GRE or ERSPAN headers. To terminate destination MAC in the packet, the packet is checked for IPv4 address and type of encapsulation. If source IPv4 address and configured encapsulation type match, an intermediate SAP is assigned to the packet and the packet is marked for dropping. The intermediate SAP is matched along with tunnel or encapsulation type and tunnel ID. If all the parameters match, it is considered a successful termination and a network SAP is derived from it. The packet is marked for decapsulation of transport header, and as the transport header is removed, the scope is automatically shifted to the next header in the packet.

The transport tunnel terminated packets are classified according to non-transport tunnel parameters configured as part of the ingress group. Based on ingress group configurations, the inner header is terminated, scope is shifted to the next header, or packets are classified and SAP is updated in the packet metadata to inner SAP (tunnel SAP). For transport packets, both transport (transport header termination) and non-transport configuration (decapsulate outer header and/or scope-shift operations and/or non-transport tunnel classification) are applied and packet classification is done based on both configurations.

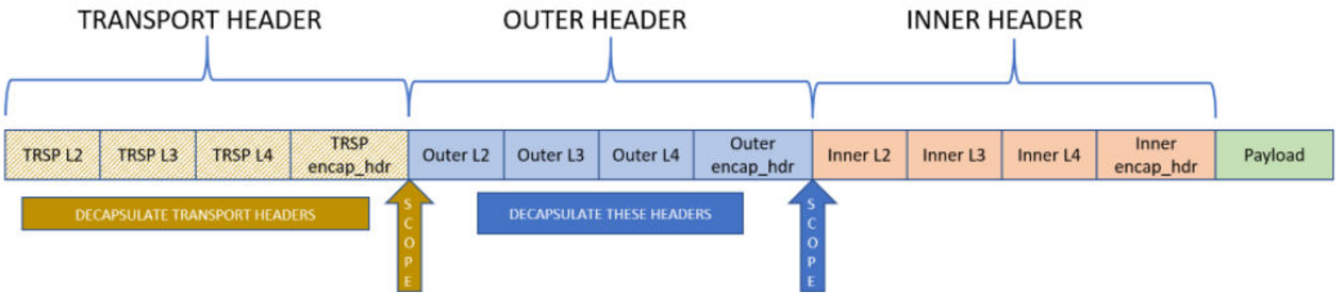


Figure 15: Transport tunnel termination in terminate mode

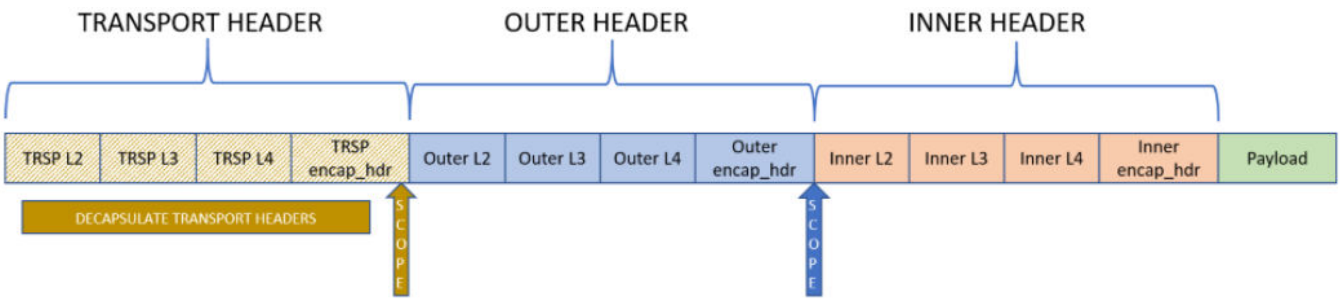


Figure 16: Transport tunnel termination in scope-shift mode

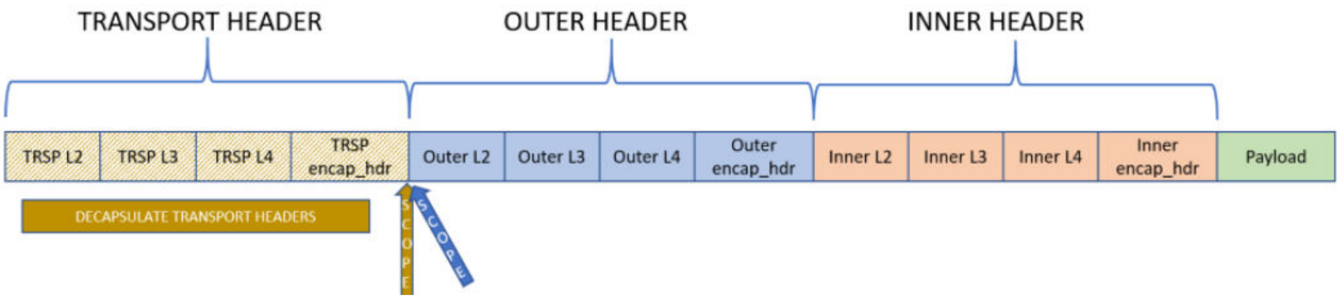


Figure 17: Transport tunnel termination in no-op mode

Terminate Non-Transport Tunnels

About This Task

Perform this procedure to terminate non-transport tunnels.

Procedure

1. Create an ingress-group with the required parameters.

```
device(config) # ingress-group name
device (config-ingress-group)#
```

2. Attach the ingress group to the required ingress port/interface.

```
device(config)# interface ethernet 1/12
device(config-if-eth-0/12)# set ingress-group name
```

3. Configure the required traffic-type parameter and mode for the ingress-group.

    - Terminate mode: Decapsulates the configured packet headers.
    - Scope-shift mode: Shifts scope of the configured packets to inner headers.

    ```
    device(config-ingress-group)# traffic-type {gre | gtpu | ipip | nvgre | vxlan} mode
    [decap | new-scope]
    ```

    - The outer (`sap-id`) and inner (`inner-sap-id`) tunnel SAP IDs are generated for tunnel levels one and two.
    - If the mode to decapsulate or terminate packet headers is not specified, packets are classified based on configured traffic type and tunnel ID parameters.

    > Note
    >
    > The outer SAP ID is not used as only one level of tunnel termination or decapsulation is supported in the NPB application.

4. Configure L3-ACL to forward the traffic.

    a. Set ACL in the route-map.

    b. Bind route-map to the ingress-group.

    ```
    device(config)# ip access-list name
    device(config)# route-map name sequence_number
    device(config-route-map) # match {mac | ip | ipv6} name
    device(config-route-map-name) # forward-action permit
    ```

5. Configure the required ports, interfaces, or port-channel:

    - List of egress ports/interfaces in egress-group and associated group in route-map.
    - Egress interface in route-map.
    - Port-channel in egress-group and associated group in route-map.

    ```
    device(config) # interface ethernet slot/port
    device(config-if) # set ingress-group name
    ```

Terminate Transport Tunnels

About This Task

Perform this procedure to terminate GRE/nvGRE/VXLAN/IPIP/GTPu packet headers from incoming packets.

Procedure

1. Create a transport tunnel with the required parameters.

    ```
    device (config) # transport-tunnel name
    tunnel-type [ gre | erspan ] [ src-ip ipaddr | mask mask ] [tunnel-id value ]
    ```

2. Create an ingress-group with the required parameters.

    ```
    device (config) # ingress-group name
    ```

3. Configure the required traffic-type parameter and mode for the ingress-group.

   - Terminate mode: Decapsulates IPGRE/nvGRE/VXLAN/IPIP/GTPu packet headers.
   - Scope-shift mode: Shifts scope of IPGRE/nvGRE/VXLAN/IPIP/GTPu packets to inner headers.

   ```
   device (config-ingress-group) # traffic-type {gre | gtpu | ipip | nvgre | vxlan} mode
   [decap | new-scope]
   ```

   - The outer (`sap-id`) and inner (`inner-sap-id`) tunnel SAP IDs are generated for tunnel levels one and two.
   - If the mode to decapsulate or terminate packet headers is not specified, packets are classified based on configured traffic type and tunnel ID parameters.

   Note

   The outer SAP ID is not used as only one level of tunnel termination or decapsulation is supported in the NPB application.

4. Attach the ingress group to the required ingress port/interface.********?????

   ```
   device (config-ingress-group)# set ingress-group name
   ```

5. Attach the ingress group to the transport tunnel.

   ```
   device(config)# transport-tunnel name
   device(config-transport-tunnel)# set ingress-group name
   ```

6. Configure L3-ACL to forward the traffic:

   a. Set ACL in the route-map.

   b. Bind route-map to the ingress-group.

   ```
   device(config)# ip access-list name
   device (config)# route-map name sequence_number
   device (config-route-map) # match ip name
   device (config-route-map-name) # forward-action permit
   ```

7. Configure the required ports, interfaces, or port-channel:

   - List of egress ports/interfaces in egress-group and associated group in route-map.
   - Egress interface in route-map.
   - Port-channel in egress-group and set the group in route-map.

   ```
   device (config) # interface ethernet slot/port
   device (config-if) # set ingress-group ingress-group-name
   ```

View Transport Tunnel Statistics

About This Task

Counters for non-transport tunnel type ingress-group is not supported.

Procedure

1. View transport tunnel configuration.

   ```
   # show transport-tunnel [ all | name ]
   ```

2. View transport tunnel counters information.

   ```
   # show counters transport-tunnel name
   ```

Terminate and Scope Shift Matrix

The following table summarizes the behavior of scope-shift and terminate in the following stages of packet processing:

- Ingress Group
- Route Map
- Listener Policy

Note

- Scope-shift is not supported simultaneously in two stages.
- If scope-shift comes before a terminate, the terminate action terminates two layers of the tunnel.

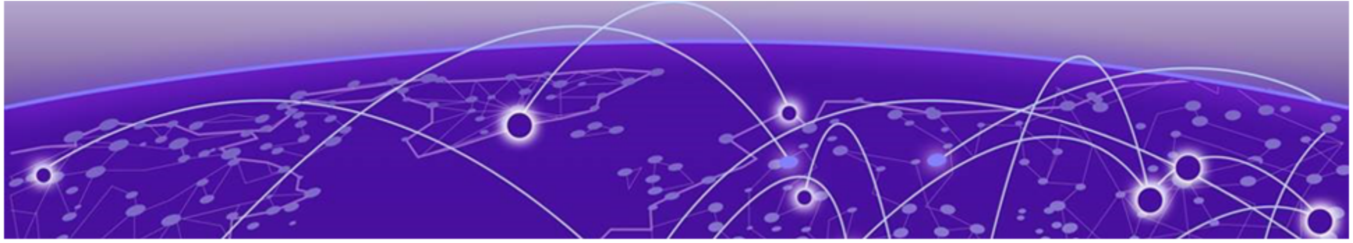
Packet Capture

```
Ethernet II, Src: 00:00:00_00:00:02 (00:00:00:00:00:02), Dst: 00:00:00_00:00:01
(00:00:00:00:00:01)
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 100
Internet Protocol Version 4, Src: 1.1.1.1, DST: 7.7.7.7        1st GRE Layer
Generic Routing Encapsulation (IP)
Internet Protocol Version 4, Src: 2.2.2.2, DST: 8.8.8.8        2nd GRE Layer
Generic Routing Encapsulation (IP)
Internet Protocol Version 4, Src: 3.3.3.3, DST: 9.9.9.9        3rd GRE Layer
Generic Routing Encapsulation (IP)
Internet Protocol Version 4, Src: 4.4.4.4, DST: 10.10.10.10
Data (886 bytes)
```

#	Ingress Group	Route Map	Listener Policy	Expected Behavior
1	Terminate	Scope-shift	Terminate	• Ingress Group terminates 1st GRE layer. • Route Map lookup happens with 2nd IP layer. • Route map shifts scope to 3rd IP layer. • Listener policy lookup happens with 3rd IP Layer. • 2nd and 3rd GRE layer is terminated by listener policy.
2	Terminate	Scope-shift		• Ingress Group terminates 1st GRE layer. • Route Map lookup happens with 2nd IP layer. • Route map shifts scope to 3rd IP layer. • Listener policy lookup happens with 3rd IP Layer.
3	Terminate	Terminate	Terminate	• Ingress Group terminates 1st GRE layer. • Route Map lookup happens with 2nd IP layer. • Route map terminates 2nd GRE layer. • Listener policy lookup happens with 3rd IP Layer. • Listener policy terminates 3rd GRE Layer.

#	Ingress Group	Route Map	Listener Policy	Expected Behavior
4	Terminate	Terminate		• Ingress Group terminates 1st GRE layer. • Route Map lookup happens with 2nd IP layer. • Route map terminates 2nd IP layer. • Listener policy lookup happens with 3rd IP Layer.
5	Terminate		Terminate	• Ingress Group terminates 1st GRE layer. • Route Map lookup happens with 2nd IP layer. • Listener policy lookup happens with 2nd IP Layer. • Listener policy terminates 2nd GRE Layer.
6	Scope-shift	Scope-shift	Terminate	Scope-shift is not supported simultaneously in two stages.
7	Scope-shift	Scope-shift		Scope-shift is not supported simultaneously in two stages.
8	Scope-shift	Terminate	Terminate	• Ingress Group shifts scope to 2nd IP layer. • Route Map lookup happens with 2nd IP layer. • 1st and 2nd GRE layer is terminated by route map. • Listener policy lookup happens with 3rd IP Layer. • 3rd GRE layer is terminated by listener policy.
9	Scope-shift	Terminate		• Ingress Group shifts scope to 2nd IP layer. • Route Map lookup happens with 2nd IP layer. • 1st and 2nd GRE layer is terminated by route map. • Listener policy lookup happens with 3rd IP Layer.
10	Scope-shift		Terminate	• Ingress Group shifts scope to 2nd IP layer. • Route Map lookup happens with 2nd IP layer • Listener policy lookup happens with 2nd IP Layer. • 1st and 2nd GRE layer is terminated by listener policy.
11		Scope-shift	Terminate	• Route Map lookup happens with 1st IP layer. • Route map shifts scope to 2nd IP layer. • Listener policy lookup happens with 2nd IP Layer. • 1st and 2nd GRE layer is terminated by listener policy.
12		Scope-shift		• Route Map lookup happens with 1st IP layer. • Route map shifts scope to 2nd IP layer. • Listener policy lookup happens with 2nd IP Layer. • No change in Packet.

#	Ingress Group	Route Map	Listener Policy	Expected Behavior
13		Terminate	Terminate	• Route Map lookup happens with 1st IP layer. • 1st GRE layer is terminated by route map. • Listener policy lookup happens with 2nd IP Layer. • 2nd GRE layer is terminated by listener policy.
14		Terminate		• Route Map lookup happens with 1st IP layer. • 1st GRE layer is terminated by route map. • Listener policy lookup happens with 2nd IP Layer.
15			Terminate	• All Lookup happens with 1st IP layer. • 1st GRE layer is terminated by listener policy.
16				• All Lookup happens with 1st IP layer. • No change in egress packet.



L2 Protocol Services

The topics in this section describe L2 protocol services.

L2 Protocol Services

L2 Protocol Service modules can be integrated into NPB application microservices based architecture using Standard OpenConfig/YANG models. For more information on YANG, see *Extreme 9920 Software YANG Reference Guide, 21.1.0.0*.

gNMI is a framework for network device management that uses gRPC as the transport mechanism. NPB application uses gRPC Transport mechanism based on ProtoBuf messages for communication across various microservices. For more information on gNMI/API, see gRPC API Gateway on page 81.

The following figure shows an high level architecture of L2 Protocol Service.

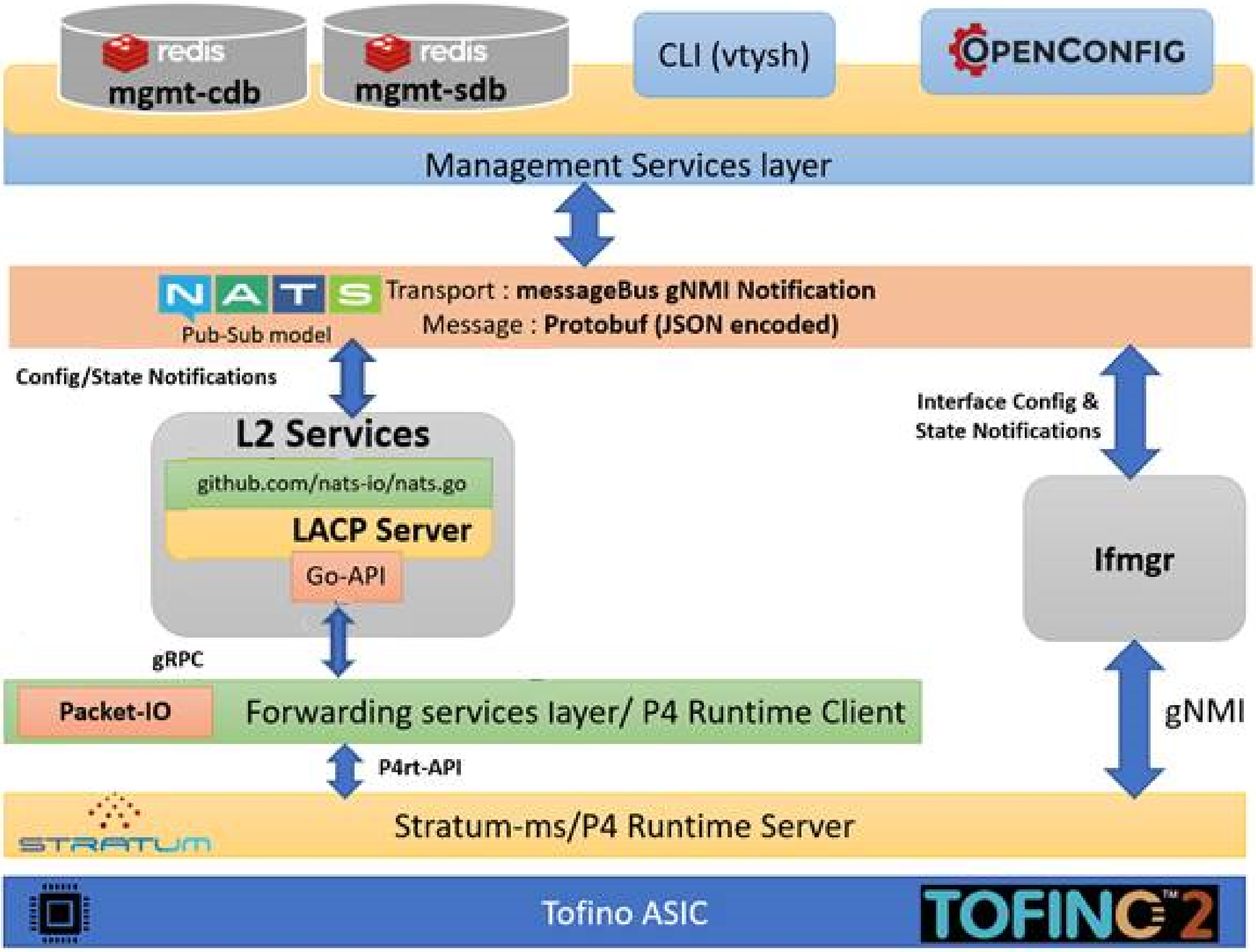

redis
mgmt-cdb
redis
mgmt-sdb
CLI (vtysh)
OPENCONFIG
Management Services layer
NATS
Pub-Sub model
Transport : messageBus gNMI Notification
Message : Protobuf (JSON encoded)
Config/State Notifications
Interface Config &
State Notifications
L2 Services
github.com/nats-io/nats.go
LACP Server
Go-API
Ifmgr
gRPC
Packet-IO
Forwarding services layer/ P4 Runtime Client
gNMI
P4rt-API
STRATUM
Stratum-ms/P4 Runtime Server
Tofino ASIC
TOFINO 2

Figure 18: High level architecture of L2 protocol service

OpenSource L2 Protocol Stack

Following are the main features of OpenSource L2 Protocol Stack:

- Go - language based
- Licensing based on Apache-2.0
- Thrift RPC and RESTAPI based interfaces
- Modular code can be replaced by gRPC and Open Config/YANG
- STP modules (not supported in NPB application)

Link Aggregation Overview

Link Aggregation (LAG) bundles multiple physical Ethernet links into a single port-channel for enhanced performance and redundancy.

LAG provides load balancing across physical interfaces and improves reliability. The port-channel stays operational as long as at least one physical interface within the port-channel is operational.

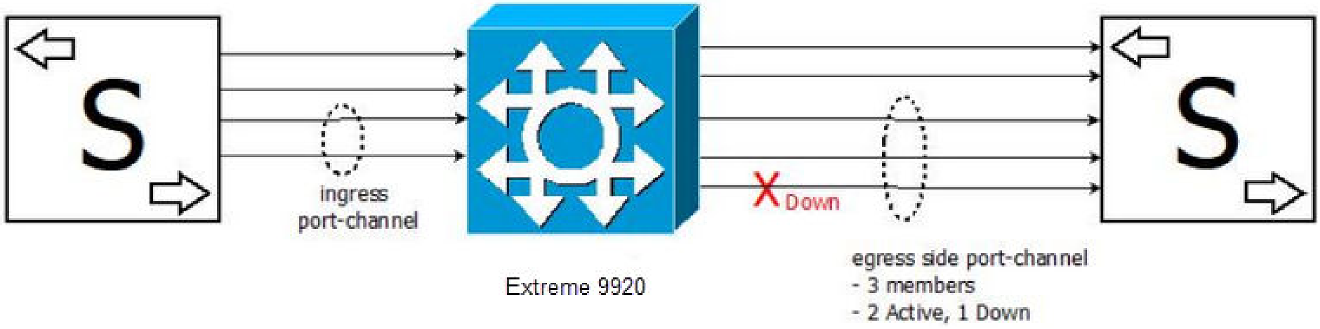


Figure 19: LAG or port-channel

LAG uses various system components such as messaging infrastructure, L2 Protocol Services, Forwarding Services, and Stratum.

Each LAG consists of the following components:

- Links of the same speed
- A MAC address that is different from the MAC addresses of the individual LAG member links.
- An interface index for each link for neighboring devices to identify.
- An administrative key for each link. Only the links with the same administrative key value can be aggregated into a LAG.

NPB application supports Static (manual) LAG.

Static LAG

In static link aggregation, administrator can create port-channel interface or LAG and add member interfaces manually using CLI commands or gNMI set requests.

In static link aggregation, Link Aggregation Control Protocol (LACP) packets are not exchanged between the partner systems. Aggregation and load-balancing of frames on static links is determined by the operational status and administrative state of the link.

Min-Links define the operational state of a LAG interface. If the number of operationally up Ethernet ports are less than configured min-links value, the LAG interface is considered operationally down. By default, min-links value is set to 1. At least one member port must be up, for a LAG interface to be operationally up.

Note

- Member ports in channel-group can be added or deleted without creating the port-channel.
- Updating state DB and creating hardware entry can be triggered only after creating the port-channel.

Traffic Aggregation

Traffic received on multiple ingress interfaces or Test Access Points (TAPs) is aggregated, filtered, and forwarded to a monitoring tool on egress interface or egress-group. The forwarding decision is based on the ACLs and route-maps applied on the aggregated logical interface or port-channel.

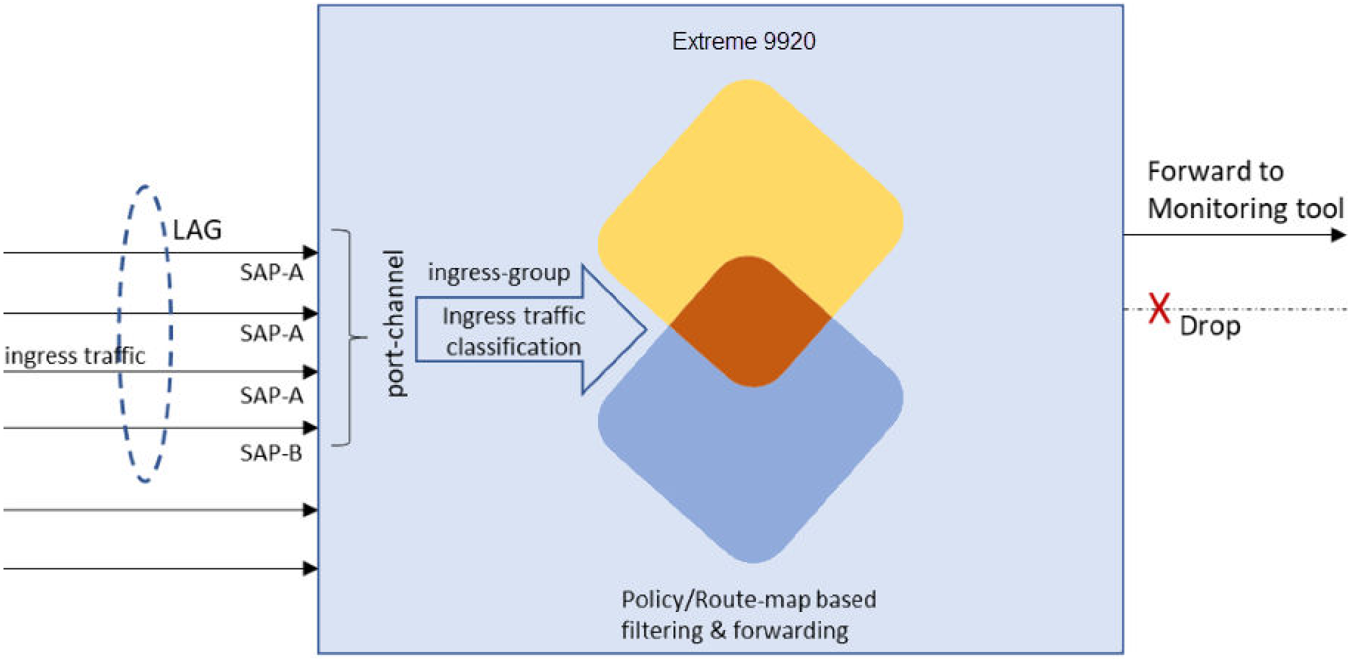

Extreme 9920
LAG
SAP-A
SAP-A
ingress traffic
SAP-A
SAP-B
port-channel
ingress-group
Ingress traffic classification
Forward to Monitoring tool
X
Drop
Policy/Route-map based filtering & forwarding

Figure 20: Traffic aggregation and packet forwarding

Configure Static LAG

About This Task

Perform this procedure to configure a static LAG.

Procedure

1. Enter global configuration mode and configure the LAG or port-channel with single or multiple interfaces.

```
device# configure terminal
device(config)# interface port-channel 1
```

2. Enter interface configuration mode and add LAG interfaces or port-channels to ingress (LAG1) and egress (LAG2) groups, as required.

   An interface cannot be part of two LAGs.

   ```
   device(config)# interface ethernet 1/1
   device(config-if)# channel-group LAG1 mode on
   device(config-if)# channel-group LAG2 mode on
   ```
3. Configure the required options for L2 (MAC) or L3 (IP/IPv6) ACL in the access list:
   - L2 (MAC) ACL:

     ```
     # match mac access-list name
     ```
   - L3 (IP) ACL:

     ```
     # match ip access-list name
     ```
   - L3 (IPv6) ACL:

     ```
     # match ipv6 access-list name
     ```
4. Enter route-map configuration mode and configure L2 or L3 ACL in the route-map.
   - Deny: Traffic flows or packets that match this criteria are dropped.
   - Permit: Traffic flows or packets that match this criteria are forwarded to egress group or egress port.

   ```
   device# config-route-map
   device(config-route-map)# forward-action [ permit | deny ]
   ```

   The traffic flows or packets are dropped or forwarded based on the configured criteria in the route-map.
5. Apply route-map policy to the ingress group.

   ```
   device (config-ingress-group) # set route-map name
   ```
6. Send the traffic flows to the DUT.
7. Verify that the traffic flows received from multiple interfaces of LAG1 is aggregated.
8. Verify that the forwarded traffic received at LAG2 is symmetric load balanced across multiple egress ports.
9. Verify the CLI LAG/port-channel statistics to determine the number of flows aggregated and/or load balanced.
10. Verify the CLI ingress and/or egress group statistics to determine the number of packets or flows dropped.

Load Balancing

Based on policy decision, NPB application balances egress traffic load across all active physical interfaces of a port-channel or LAG. Load balance algorithm is based on symmetric hash generated by L2 MAC addresses for L2 packets and 5 tuple parameters for L3 packets.

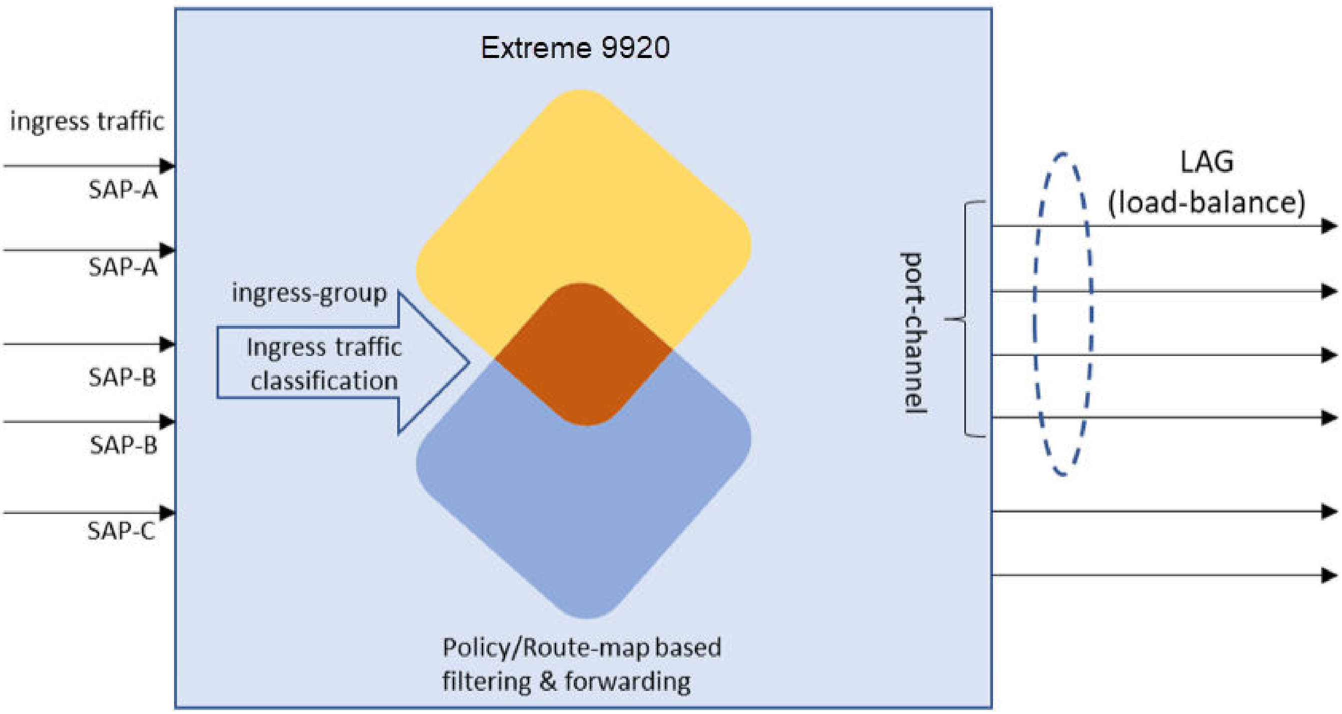


Figure 21: LAG load balancing

Note

- A physical interface can be part of only one port-channel or LAG.
- A small drop in traffic can be experienced when an active interface fails.

Enable Load Balancing

About This Task

NPB application supports GTP tunnel ID based load balancing.

Procedure

1. Enter interface port-channel configuration mode.

```
Device# configure terminal
Device#(config)# interface port-channel 1
Device(config-if-po-1)#
```

2. Enable load balancing.

```
Device(config-if-po-1)# load-balance src-dst-ip-l4port
```

Note
For tunnel ID based load balancing, go to step 3.

3. Enable tunnel ID based load balancing.

```
Device(config-if-po-1)# load-balance src-dst-ip-l4port-tid
```

SNMP Monitoring

Extreme 9920 software implements Simple Network Management Protocol (SNMP) for monitoring system events. SNMP traps send alert messages from a remote SNMP-enabled device to a central collector. SNMP support in the application implements the open-source net-snmp protocol stack, in which a single 'agentx subagent' container fetches requested MIB data and feeds it to the net-snmp protocol stack. This is shown in the following figure.

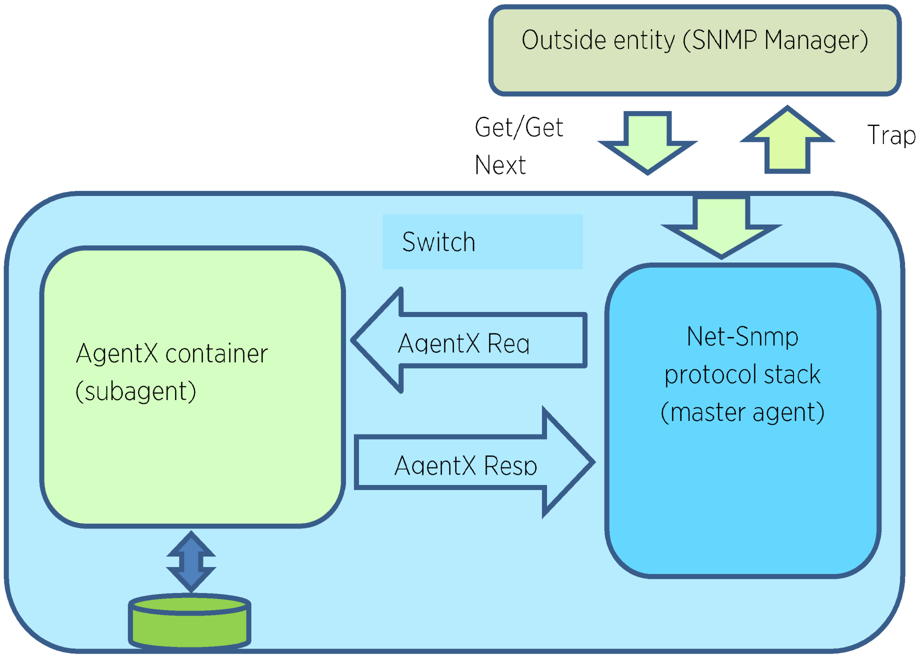

Outside entity (SNMP Manager)
Get/Get Next
Traps
Switch
AgentX container (subagent)
AgentX Req
AgentX Resp
Net-Snmp protocol stack (master agent)

Figure 22: net-snmp protocol stack

You can configure SNMP to collect the following information:

- IfEntry
- IfXEntry
- system
- entPhysicalEntry

Supported SNMP Traps

Learn about the SNMP traps that are supported in Extreme 9920 software.

SNMP traps are alert messages sent from a remote SNMP-enabled device to a central collector, called the SNMP manager. In more technical terms, SNMP traps are asynchronous, unpacked messages used to notify an entity in your network management system of significant issues and events.

Extreme 9920 software supports the following generic SNMP traps.

Table 7: Supported SNMP Traps

Name	Description	When initialized
coldStart	A coldStart trap signifies that the sending protocol entity is reinitializing itself such that the agent's configuration or the protocol entity implementation may be altered.	After the device is restarted. After the SNMP service on the device is restarted.

linkDown	A linkDown trap signifies that the sending protocol entity recognizes a failure in one of the communication links represented in the agent's configuration.	A linkDown trap is activated whenever the admin status of any interface is changed from UP to DOWN .
		Example:
		`device(config)# interface ethernet 1/2`
		`device(config-if-eth-1/2)# shutdown`
		The **`shutdown`** command initializes a linkDown trap.

linkUp	A linkUp trap signifies that the sending protocol entity recognizes that one of the communication links represented in the agent's configuration has come up.	A linkUp trap is activated whenever the admin status of any interface is changed from DOWN to UP.
		Example:
		`device(config)# interface ethernet 1/2`
		`device(config-if-eth-1/2)# no shutdown`
		The **`no shutdown`** command initializes a linkUp trap.

Configure the SNMP Community String

Learn how to configure a community string to send notification of significant system issues and events.

About This Task

You configure an SNMPv2 community string community string (SNMP string) to access device information via SNMP. It is important to configure the string correctly; otherwise the request for device and system information will be ignored and fail silently.

- All SNMP access is read-only.
- SNMP v1 and v2c is the default.

Procedure

1. Run the **`configure terminal`** command to access Config mode.

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Run the command `snmp-server community community-name` to configure the SNMP v2 string, replacing the variable with the name of the community to be monitored.

   ```
   device# snmp-server community extreme03
   ```

3. Exit configuration mode and run the `show snmp server` command to verify that the SNMP community you configured is listed with all supported communities in the output, as shown in the following example.

   ```
   device# show snmp server
   snmp-server community extreme03
   ```

Configure an SNMP-Trap Receiver IP Address

Learn how to configure the agent with the trap-destination information.

Before You Begin

This command is available only to users with admin role.

About This Task

You configure an agent with the SNMP-trap destination information and the community. For a given host/community combination, you can enable either SNMPv1 or SNMPv2 traps, and you can configure nor more than 12 hosts for a given community.

Procedure

1. Run the **`configure terminal`** command to access Config mode.

   The command line changes to configuration mode.

   ```
   device(config)#
   ```

2. Run the `snmp-server host` and optional parameters as needed for your purposes, as shown in the following example.

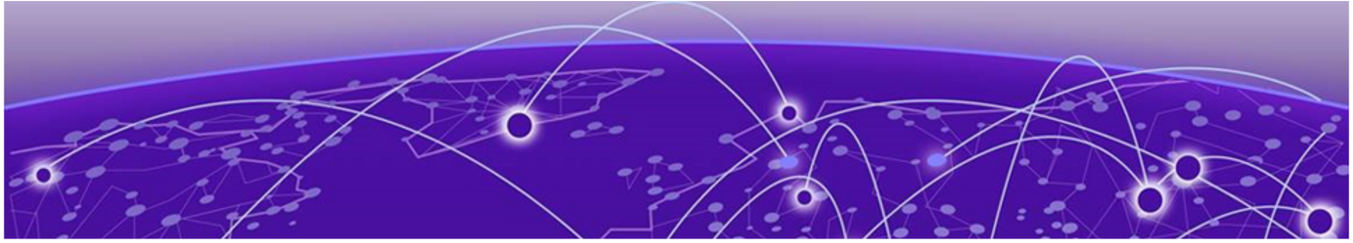
   ```
   device(config)# snmp-server host 10.23.17.128 public 162 version 2c
   ```

Monitor Events with SNMP

Learn about the resources available for configuring a monitoring station.

About This Task

- For information about monitoring events using SNMP get, getNext, and walk, refer to gRPC API Gateway on page 81.
- For information about supported MIBs, refer to the *Extreme 9920 Software MIB Reference, 21.1.0.0*.
- Default SNMP MIB files are stored at /usr/share/snmp/mibs/ on the device.



Platform and Infrastructure Services

The following topics describe platform and infrastructure services.

Interface Manager Service

Interface Manager Service in TierraOS is responsible for configuring and monitoring Ethernet Interfaces on the Switch. TierraOS is built on Stratum which provides a forwarding device or silicon agnostic API for managing the data plane, interfaces/ports, and other platform elements. Interface Manager interacts with Stratum for interface management.

Interface Manager Service provides the following features:

- Processing platform notifications (chassis ready, line card insertion or deletion)
- Publishing interface creation, deletion, or state-change notifications to clients
- Processing interface config notifications from management services
- Interacting with Stratum through gNMI APIs to monitor and configure physical ports or interfaces.
- Configuring management interface with IP address and IP gateway.

Note

Management interface supports only IPv4 address configuration.

Maximum Transmission Unit

Maximum Transmission Unit (MTU) is the size of the largest packet that can be sent over a network. The size of the MTU determines the amount of data that can be transmitted.

If a packet is larger than the MTU allowed by the frame, the device fragments the IP packet into multiple parts that fit into frames, and sends the parts of the fragmented IP packet separately, in different frames. The device that receives the multiple fragments of the IP packet reassembles the fragments into the original packet. The packet fragmentation can be reduced by increasing the MTU.

Extreme 9920 supports jumbo packets of 9000 bytes. You can change the MTU for individual IP interfaces. MTU can be set on both global and interface levels. The default interface MTU is set to 9216 and it can be changed at the interface level. The valid range is 1024-9216. The global MTU applies to all interfaces. The interface MTU configuration overrides global MTU configuration.

LAG Manager

LAG Manager is part of the Interface Manager.

LAG Manager handles the application Data Plane LAG Member tables programming based on the LACP Service events:

- Create or delete LAG
- Map LAG-Index in HW
- Add or remove active member port information in LAG Member tables

When LACP module publishes the port-channel creation and active member list, LAG manager calls the respective `p4runtime` APIs and installs the hardware.

For more information on gNMI APIs, see gRPC API Gateway on page 81.

Management Services

Management Services infrastructure is a suite of modular services. Each service is scalable and can be deployed independently. Management Services supports polyglot programming, messaging, and persistence infrastructure.

On system boot-up or Management Services restart, as part of the initialization, it sets up various DB and messaging related services.

Management Services infrastructure provides the following services:

- Database Service for storing configuration and operational state information
- API gateway supporting gNMI for run-time configurations and monitoring and gNOI protocols for run-time operations

  For more information on gNMI API, see gRPC API Gateway on page 81.
- CLIs for adding plug-in code to render show commands and process configuration commands

  For more information on Extreme 9920 software commands, see *Extreme 9920 Software Command Reference, 21.1.0.0*.

The following figure shows an overview of management services.

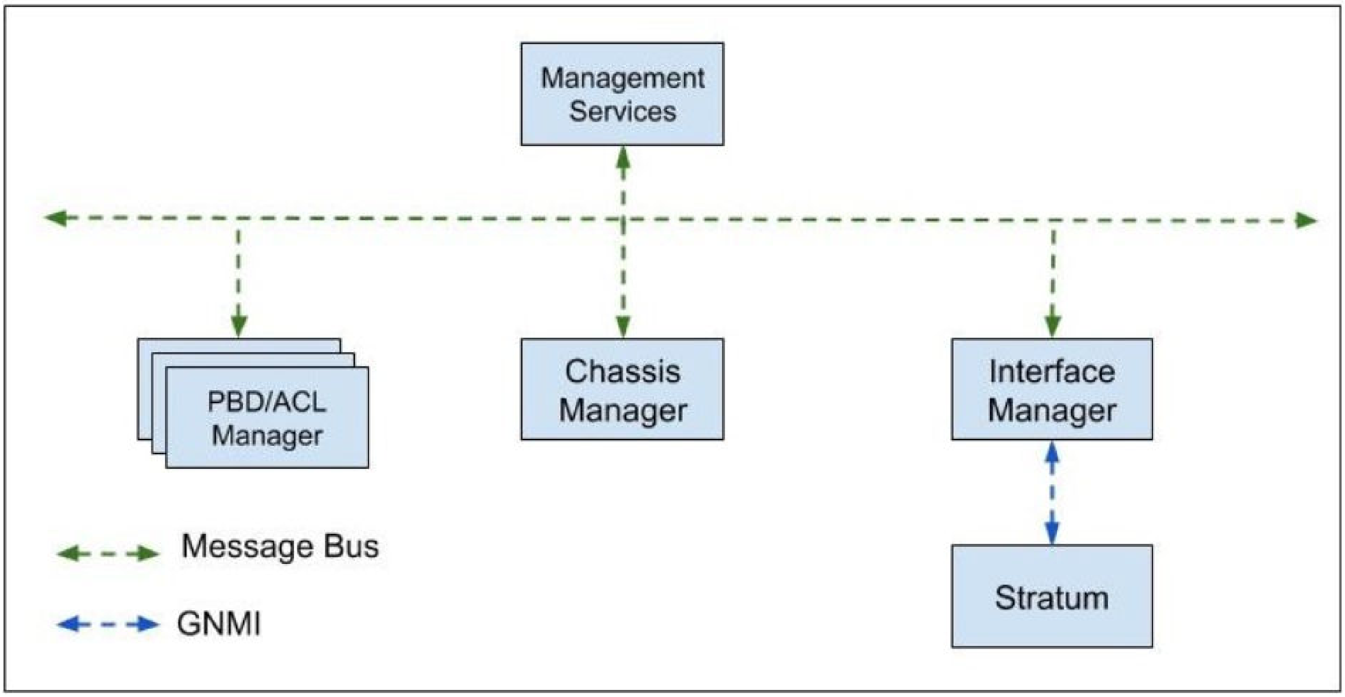


Figure 23: Overview of management services

Command Line Interface

Command Line Interface (CLI) processes commands and helps configure and manage the device.

CLI acts as a GNMI client for configuration commands or a database client to fetch data and render show command output.

For information on Extreme 9920 software commands, see *Extreme 9920 Software Command Reference, 21.1.0.0*.

DB Services

DB Service is responsible for defining and managing various message topics.

DB Service acts as a central repository for both state and config data.

DB Service is the primary consumer of all config/state notifications from feature services. DB Service handles necessary conversion and writes or reads from the appropriate internal data store to deliver it to the requester. Persisted data is hierarchical in nature, per OpenConfig Yang modules.

Running Configuration

NPB application configuration is persistent after device reboot. Upon device reboot, Extreme 9920 is programmed per persisted configuration.

The configuration currently effective on the device is referred to as the running configuration. Any configuration change you make while the device is online is made to the running configuration.

Apply Configuration

Before You Begin

Before upgrading or downgrading the firmware, backup the running configuration.

Procedure

1. Save running configuration to a file.

   ```
   device# copy running-config flash://config-file/file-name
   ```

2. View configuration files.

   ```
   device# show flash://file-type/file-name
   ```

3. Apply configuration from a saved config file.

   ```
   device# copy flash://config-file/file-name running-config
   ```

Reset Configuration

Before You Begin

Before upgrading or downgrading the firmware, backup the running configuration.

Procedure

Save default configuration to running configuration to reset device configuration.

```
device# copy default-config running-config
```

Managing Flash Files

Extreme devices provide a set of tools for removing, renaming, and displaying files that are created in the device flash memory. You can use the display commands with any file, including the system configuration files. The rename and delete commands only apply to copies of configuration files you create in the flash memory. You cannot rename or delete any of the system configuration files.

Table 8: Managing flash files

Command	Task
dir flash:// [chassis-ms | config-file | coredumps | ifmgr-ms | lacp-ms | lldp-ms | mgmt-cli | mgmt-snmp-agent | mgmtsvc-apigw | ms_images | pcap-file]	Lists contents of the flash memory.
show flash://file-type/file-name	View contents of the file in flash memory.
delete flash://file-type/file-name	Deletes file from the flash memory.

Chassis Manager

Chassis manager is responsible for managing hardware modules and maximizing the availability of the services hosted on the chassis. Chassis Manager is platform independent and depends on the ONLP Platform abstraction layer.

Interface Manager can subscribe to Chassis Manager to receive and process platform notifications such as Chassis Ready, Line Card Insertion, or Deletion. The port configuration is pushed to stratum to configure the ports in the hardware based on the Line Card type.

The ONLP layer consists of a Platform independent layer (ONLP-PI) and a Platform-dependent layer (ONLP-VM) modules. The platform dependent modules abstract the underlying hardware through ONLP interfaces. Chassis manager runs as a docker container and consists of the following sub-modules:

- Chassis Worker - Manages core chassis functionality.
- PSU Worker - Manages power-supply units and reports state of the PSUs to OperDB.
- Fan Worker - Manages fans and alerts if any fan unit fails.
- Slots Worker - Monitors slots in a pluggable chassis and handles state changes such as line card insertion or removal.
- Health Monitor - Monitors chassis health, including CPU usage, memory usage, and temperature.
- CLI Server - Provides connectivity to the chassis for debug.

Stratum

Stratum is an open source silicon-independent switch operating system for software defined networks. It provides P4Runtime APIs for data plane Control (programming forwarding behavior and forwarding tables) and Openconfig based gNMI APIs for Configuration and Monitoring of Switch elements such as Ports, QoS, Platform that are not programmable by P4Runtime. It does not embed any control protocols and is designed to support an embedded or external Network Operating System.

For more information on gNMI API, see gRPC API Gateway on page 81.

Stratum builds an Openconfig YANG path tree based on the Chassis config pushed down and subsequent gNMI Get/Set/Subscribe requests are supported only for these paths.

For more information on YANG, see *Extreme 9920 Software YANG Reference Guide, 21.1.0.0*

Line Card

Chassis Manager detects Line Card (LC) insert or remove events from ONLP and publishes these events along with the line card type and slot number information. The port default properties for the line card type is published in the line card specific config file.

Interface Manager updates the chassis configuration file when LC insert or removal event is received and notifies stratum of the port map change using GNMI. Stratum processes the difference in the new config and initializes the new ports.

Interface Manager is also responsible for updating the operDB with the new interface create events and pushing the configuration for these interfaces down to stratum. Interface Manager monitors the state of the new interfaces and updates the state to operDB.

Port Management

NPB application allows port management of the Extreme 9920-NPB-8 and 9920-16C devices.

Configure Breakout Mode

Procedure

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter hardware configuration mode.

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

3. Configure the required breakout mode.

   - `4x10g`
   - `4x25g`

   ```
   device (config-hardware)# breakout mode [ 4x10g | 4x25g ]
   ```

Port Creation

After a cold reboot or firmware upgrade, Interface Manager waits for the following events to complete to create ports on the system.

- Chassis Manager to detect and publish the chassis type to `operDB`.
- Stratum to initiate configuration management and P4 services, reconfigure ports using the chassis config file, and start Stratum gNMI service.

Interface Manager performs the following tasks to create ports:

1. Connects to stratum and creates interfaces based on the chassis type and configuration.
2. Queries user configurations from state DB and pushes them to stratum.
3. Subscribes to operational state updates from stratum for each of the ports.
4. Updates `operDB` with configuration and state of the ports.

Network Time Protocol

Network Time Protocol (NTP) maintains uniform time across all devices in a network. The NTP commands support the configuration of an external time server to maintain synchronization among all local clocks in a network.

Date and Time Settings

Extreme devices maintain the current date and time inside a battery-backed real-time clock (RTC) circuit. Date and time are used for logging events. Device operation does not depend on the date and time; a device with incorrect date and time settings can function correctly. However, because the date and time are used for logging, error detection, and troubleshooting, you should set them correctly.

Time Zone Settings

The time zone settings have the following characteristics:

- The setting automatically adjusts for Daylight Savings Time.
- Changing the time zone on a device updates the local time zone setup and is reflected in local time calculations.
- System services that have already started will reflect the time zone changes only after the next reboot.
- Time zone settings are not affected by NTP server synchronization.

Network Time Protocol Server

Network Time Protocol (NTP) server provides the correct network time on the device. NTP can be used to synchronize the time on devices across a network.

Network Time Protocol server is used to obtain the correct time from an external time source and adjust the local time in each connected device. When NTP server functionality is enabled, the NTP server starts listening on the NTP port for client requests and responds with the reference time. Up to five server addresses in IPv4 format can be configured. When you configure multiple NTP server addresses, the first obtainable address is set as the active NTP server. If there are no reachable time servers, then local device time is the default time until a new active time server is configured.

When synchronization is lost, NTP server operates in master mode to serve time using the local clock. Serving local clock can be enabled using the NTP master command.

The NTP server is stateless and does not maintain any NTP client information. Network time synchronization is guaranteed only when a common external time server is used by all devices.

Configure NTP

About This Task

After setting the date and time on a device, the local time on a device can be synchronized with a Network Time Protocol (NTP) server.

The date and time are set in Exec mode and only have to be configured once per device because the value is written to nonvolatile memory. After the basic time information is set up, an NTP server is configured to allow the local time to be synchronized across the network.

Procedure

1. Enable NTP.

   ```
   ntp enable
   ```

2. Disable NTP.

   ```
   no ntp enable
   ```

3. Configure NTP peer IPv4 server.

   ```
   ntp peer ipv4 address
   ```

4. Disable NTP peer IPv4 server.

   ```
   no ntp peer ipv4 address
   ```

5. Configure NTP IPv4 server.

   ```
   ntp server ipv4 address
   ```

6. Remove NTP IPv4 server.

   ```
   no ntp server ipv4 address
   ```

gRPC API Gateway

NPB application supports the Remote Procedure Calls (RPCs) defined in the gRPC Network Management Interface (gNMI) specification. Extreme 9920 supports gNMI version 0.7.0.

The complete text of the gNMI specification, `gnmi-specification.md` is available on github.com: https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md.

For information on gRPC Network Operations Interface (gNOI) supported by Extreme 9920, see *Extreme 9920 Software gNOI Reference Guide, 21.1.0.0*.

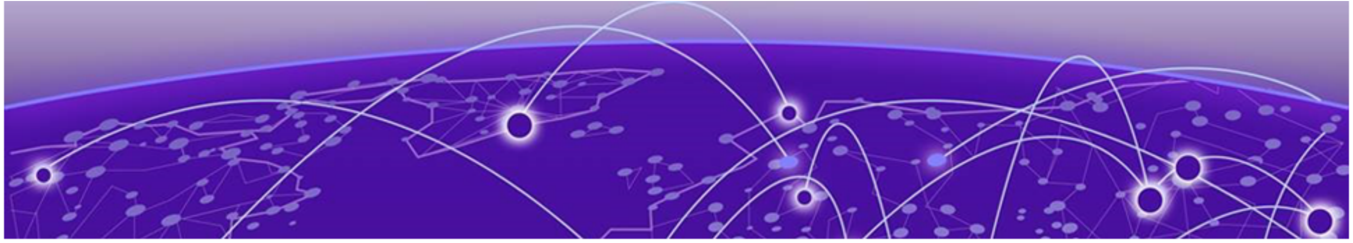
Table 9: Supported RPCs

RPC	Purpose	Section in the Specification
Capabilities	Used by the client and target as an initial handshake to exchange capability information. Consists of the following messages: • `CapabilityRequest` • `CapabilityResponse`	*Section 3.2 Capability Discovery*
Get	Used by the client to retrieve snapshots of the data on the target. Consists of the following messages: • `GetRequest` • `GetResponse`	*Section 3.3 Retrieving Snapshots of State Information*

Table 9: Supported RPCs (continued)

RPC	Purpose	Section in the Specification
Set	Used by the client to modify the state of the target. Consists of the following messages: • `SetRequest` • `SetResponse`	*Section 3.4 Modifying State*
Subscribe	Used by the client to control subscriptions to data on the target. Consists of the following messages: • `SubscribeRequest` • `SubscriptionList` • `SubscribeResponse`	*Section 3.5 Subscribing to Telemetry Updates*

These RPCs use `openconfig` YANG paths to identify the object on which create, read, update, and delete (CRUD) operations are to be performed. For more information, see the *Extreme 9920 Software YANG Reference Guide, 21.1.0.0*.



Logging

The following topics describe event logging in NPB application.

Event Logging

Management Services modules use common logging or event subsystem to log various messages and generate events (in standardized forms) to indicate progress, status change, errors, and failures.

All forwarding agents are integrated with the elog feature and support the following category of logs:

- Error
- Warning
- Info
- Debug

NPB application supports UDP, TCP, and TCP/TLS transport protocols for remote system logging. For more information, see *Extreme 9920 Software Security Configuration Guide, 21.1.0.0*.

Forward Agent Logs

Based on the event, the forwarding agents print the logs with the required information for debugging and understanding of the scenarios. Also, each log is added with the time stamp and the logs from different agents can be correlated.

The forwarding agents are integrated with the infra tracing support which indicates the functions entry and exit point to ease debugging.

All forwarding agents logs are available at `/var/log/pods/<container-specific-id>/<forwarding-agent-name>/<number>.log` where `<number>` indicates the number of times the container is restarted. The latest or highest-numbered log file must be taken for recent logs.

```
/var/log/pods/ngnpb_agent-sp-intf-ms-85cbc88bcb-cmc9j_eb47ced9-e318-4112-
b5b3-3cf5c61cff7a/agent-sp-intf-ms/25.log

/var/log/pods/ngnpb_agent-sp-nhop-ms-6db59887d4-xrq9p_429d1d16-7ddf-49d6-89d8-
f7f25e8fce25/agent-sp-nhop-ms/25.log
{"log":"\u001b[37mTRAC\u001b[0m[06:01:29.588 28-05-2020] addIngressSFPSel:
Enter
\n","stream":"stdout","time":"2020-05-28T06:01:29.58855818Z"}
```

```
{"log":"\u001b[37mTRAC\u001b[0m[06:01:29.588  28-05-2020] addIngressSFPSel:to be added
sfpsel entry {Priority:0  ControlHandle:0 Sfc:[0 13] Result1:[] Result2:[{Valid:true Spi:
[0 0 8]  Si:[6] SiPredec:[6]}]}
\n","stream":"stdout","time":"2020-05-28T06:01:29.58866029Z"}

{"log":"\u001b[37mTRAC\u001b[0m[06:01:29.600 28-05-2020] addIngressSFPSel:Added sfpsel
entry {Priority:0 ControlHandle:0 Sfc:[0 13] Result1:[]  Result2:[{Valid:true Spi:[0 0 8]
Si:[6] SiPredec:[6]}]}
\n","stream":"stdout","time":"2020-05-28T06:01:29.600646936Z"}

{"log":"\u001b[37mTRAC\u001b[0m[06:01:29.600 28-05-2020] addIngressSFPSel:
Exit
\n","stream":"stdout","time":"2020-05-28T06:01:29.600774916Z"}
```

Failure Logs

```
{"log":"\u001b[36mINFO\u001b[0m[06:46:39.799 21-05-2020] *** routemap_1 sap12 12
\n","stream":"stdout","time":"2020-05-21T06:46:39.799420893Z"} {"log":"\u001b[36mINFO
\u001b[0m[06:46:39.799 21-05-2020] ***SAP-POLICY ASSOCIATION routemap_1 sap12 12
\n","stream":"stdout","time":"2020-05-21T06:46:39.79954709Z"} {"log":"\u001b[36mINFO
\u001b[0m[06:46:39.799 21-05-2020] Route Map Binding route-map:routemap_1 serviceIF:sap12
\n","stream":"stdout","time":"2020-05-21T06:46:39.799740477Z"}
{"log":"\u001b[31mERRO\u001b[0m[06:46:39.803 21-05-2020] Ingress Entry Insert error:
\u0026\{0xc0012ac090 0xc0012aa390 0xc0012bc380 0xc001216580 0xc0012b0540 64 1 0}
; \u0026{66 0 [0 0] [0 0] [0 0 0 0] [0 0 0 0] [0 0 0 0] [0 0 0 0] [9] [255] [0] [0] [0]
[0] [0] [0] [0] [0] [0 0 0 0] [0 0 0 0] [0 12] [255 255] [0] [0] [0 0] [0 0] [0 0] [0 0]
{false}
{false} {true [0] [0] [0] [0] [0 0] [1] [0 13] [0] [0] [0] [0] [0]}}; rpc error: code =
PermissionDenied desc = Write from non-master is not permitted.; \u0026{sap12 12 3 1 true
13} {false 0} {false 0} {false 0} {false 0} {false 0} {false 0} {false 0} {false 0
{0xc0003b2bc0} 0xc004467680}
\n","stream":"stdout","time":"2020-05-21T06:46:39.803203753Z"}

{"log":"\u001b[31mERRO\u001b[0m[06:46:39.803 21-05-2020] Bind Acl IPv4-1 serviceIF:12
owner:3 direction:1 Err:1
```