ADMINISTRATION GUIDE

Extreme Virtual TAP 2.0.0 Administration Guide

9035939-00
13 February 2019

Legal Notice

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

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

Trademarks

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

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

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

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: www.extremenetworks.com/support/policies/software-licensing

Contents

Preface

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

Conventions

This section discusses the conventions used in this guide.

Notes, cautions, and warnings

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

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

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

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

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

Text formatting conventions

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

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

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

Command syntax conventions

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

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

Documentation and Training

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

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

Training

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

Getting Help

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

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

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

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

Subscribing to Service Notifications

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

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

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

4. Click **Submit**.

Providing Feedback to Us

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

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

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

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

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

Overview

Extreme Virtual TAP (vTAP) is a full-featured network visibility solution built for virtualized service provider and enterprise networks. It offers an end-to-end set of capabilities including traffic interception, filtering, load-balancing, and optimization for network monitoring and analytics tools. vTAP addresses the visibility challenge in virtual workloads.

vTAP can perform various operations such as filtering (SMARTMatch), forwarding, VLAN tag insertion/deletion, header stripping, packet slicing, sampling and IPFIX export.

NOTE
vTAP supports only ethernet packets. Other packets, such as IPX and Bluetooth, are not supported.

Policy-Based Steering

vTAP uses Policy-Based Steering (PBS), which is a method to steer traffic based on the configured policies.

Policy

A policy is a template made up of one or more rules. Each rule specifies a match-action pair. For the configured set of match criteria, the specified actions are executed.

After defining a policy, it must be applied on the interface for it to be active.

Rule

Only one rule within a policy is executed at a time even if multiple rules are successful.

A rule Id is allocated for each rule, which can be used to delete a rule. To modify a rule, first delete the rule and then add it again.

A rule is defined by either an **action** or a **match-action** pair.

Each match is a chain of **parameter-value** pairs. Each **action** is the action to be taken if the **match** is successful.

The following is the format in which a rule is defined:

```
#add rule <param1>=<value> [<param2>=<value> ... <paramN>=<value>]
  action={<Action1> | <Action 2> | <Action 3>}
  priority=<value>
```

A match must include at least one `<param>=<value>` criteria. Optionally, a chain of `<param>=<value>` can form the match criteria. In this case, each `<param>=<value>` should be successful for the match criteria of the rule to be successful.

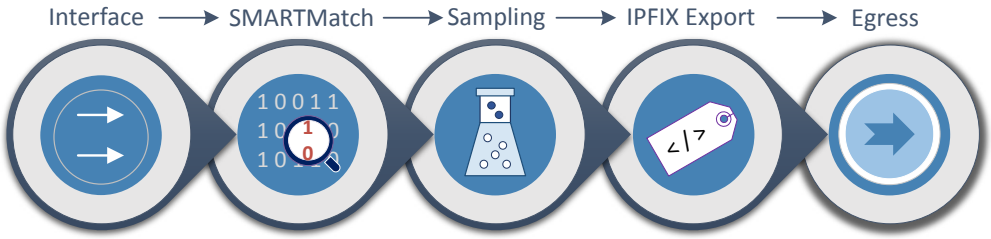
Types of policies

vTAP supports two types of policies:

- **SMARTMatch**: For information about creating a SMARTMatch policy, see the section SMARTMatch policy on page 43.
- **Sampling**: For information about creating a sampling policy, see the section Sampling policy on page 53.

vTAP supports one SMARTMatch policy and ten sampling policies. When both policies are applied on the interface, the SMARTMatch policy takes precedence over the sampling policy. A sampling policy can be an action in a SMARTMatch rule.

The following image shows the sequence in which policies and actions are executed on the interface:



NOTE

- Packets that fail ethernet decoding are sent out of the Tx vNIC, unless they are configured to be dropped. Packets that fail decoding, but have valid ethernet header are processed based on the egress configuration. In both cases, no action (SMARTMatch/sampling/IPFIX export) is applied on these packets.

  In the output for `show system-stats` command, the `Total Packets With Decode Error` counter shows the total number of packets that failed ethernet decoding.

  Example

```
vtap# show system-stats
                      Rx Packets  : 136
                        Rx Bytes  : 9864
                     Drop Packets  : 136
                       Drop Bytes  : 9864
                            Flows  : 25
                          Tunnels  : 0
                  Not Sent Flows   : 0
               Tunnels Terminated  : 0
                   Kni Rx Packets  : 136
                   Kni Tx Packets  : 0
             IP Fragmented Packets  : 128
             IP Reassembly Packets  : 32
             Total Deleted Tunnels  : 0
               Total Deleted Flows  : 0
Total Packets With Decode Error  : 40
```

- vTAP cannot recognize tunnels or flows based on only L2 header. Therefore, packets with corrupted L3 header are treated as unrecognized packets.

Packet fragmentation and reassembly

When vTAP receives fragmented packets, it reassembles these fragments to form the original packet and appropriate actions are executed on the interface as per the configuration. If the fragments arrive out of order, they are first re-ordered in the correct order and then forwarded for processing.

Incomplete packets are stored in the buffer until either all fragments are received, or the reassembly timer expires. If all the fragments are not received within the timeout period to complete the reassembly of a packet, reassembly of that packet is abandoned and all the fragments received for that packet are forwarded to the interface. No action (SMARTMatch/sampling/IPFIX export) is applied on these fragments.

NOTE

vTAP can reassemble a maximum of 10 fragments. If the number of fragments in a packet exceeds 10, it is sent to the interface without reassembly.

In the output for `show system-stats` command, the `IP Fragmented Packets` and `IP Reassembly Packets` counters show the total number of fragmented and reassembled packets respectively.

Example

```
vtap# show system-stats
                      Rx Packets  : 136
                        Rx Bytes  : 9864
                     Drop Packets  : 136
                       Drop Bytes  : 9864
                            Flows  : 25
                          Tunnels  : 0
                  Not Sent Flows   : 0
               Tunnels Terminated  : 0
                   Kni Rx Packets  : 136
                   Kni Tx Packets  : 0
            IP Fragmented Packets  : 128
            IP Reassembly Packets  : 32
            Total Deleted Tunnels  : 0
              Total Deleted Flows  : 0
Total Packets With Decode Error  : 40
```

Tunnel termination and tunnel initiation

Mirrored traffic may be forwarded in a GREv0, GREv1, VXLAN, IPIP, or ERSPAN tunnel to vTAP with the Rx VNIC IP address as the destination address. The maximum number of tunnels and flows supported is 500K.

Tunnels are terminated if:

- Destination MAC address matches the ingress MAC address (Rx)

  or

- Destination MAC address matches the ingress MAC address (Rx) and IP address matches vTAP's destination IP address

After terminating a tunnel, vTAP processes the flows/packets inside these tunnels based on the egress configuration.

NOTE

- If an incoming packet includes both MAC address and destination IP address, both have to match for the tunnel to be terminated.
- Apart from GREv0, GREv1, VXLAN, IPIP, or ERSPAN tunnels, vTAP does not support termination of any other tunnels even if the destination IP address in those packets is its own.

vTAP can also initiate GRE or VXLAN tunnels at the egress and send packets through these tunnels. A maximum of 10 egress VLAN tags or tunnels are supported.

For both GRE and VXLAN, the encapsulated packet starts from the Ethernet Header. If a VLAN or a VXLAN tunnel is configured as the egress and the encapsulated packet to be sent out does not include the Ethernet header, the Ethernet header used for VLAN or VXLAN tunnel is used for the payload.

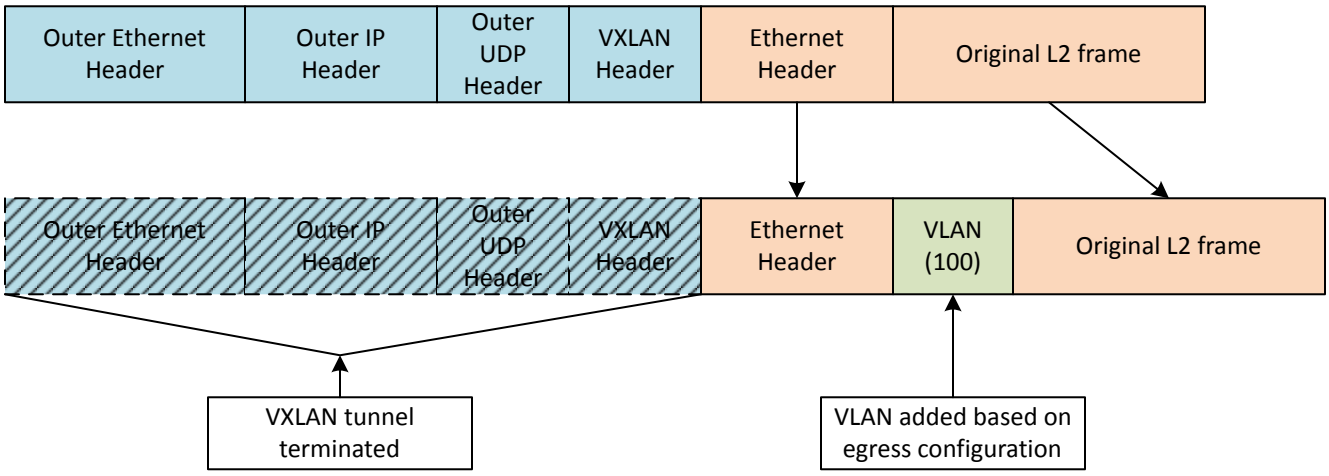
TABLE 1 Egress tunnel parameters

Parameter	Description
egress-alias	The `egress-alias` can be used in SMARTMatch rules and on the IP interface to specify the forwarding destination.
local-ip	This parameter can take any IP address as its value. It is mandatory for GRE and VXLAN tunnels. NOTE • The local IP address need not be configured on the Tx vNIC. • This parameter is optional for both IPv4 and IPv6. If it is not specified, vTAP auto-detects the value. • Both `local-ip` and `destination-ip` must be of the same type (IPv4 or IPv6).

TABLE 1 Egress tunnel parameters (continued)

Parameter	Description
destination-ip	Specifies the IP address for the tunnel destination. It is mandatory for GRE and VXLAN tunnels. NOTE • Both `local-ip` and `destination-ip` must be of the same type (IPv4 or IPv6). • The default destination port for VXLAN tunnel is 4789. This value is fixed and cannot be changed.
nexthop-mac	This parameter is mandatory for IPv6. If this value is not provided for IPv4, vTAP automatically resolves the MAC address. However, it is recommended that this parameter be specified to reduce traffic on the network.
vni	Virtual Network Identifier (VNI) on an egress-terminated VXLAN tunnel. The value can be between 1 and 16777215. This parameter is mandatory for VXLAN tunnels. NOTE The destination port for VXLAN is set to 4789 by default and cannot be changed.
tunnel-type	Specifies tunnel as either GRE or VXLAN.

VXLAN tunnel termination - Egress VLAN

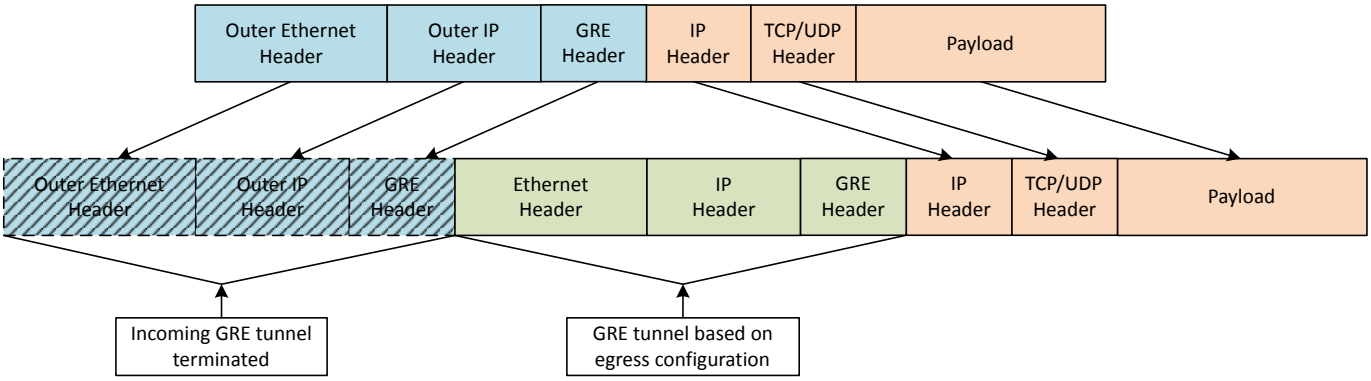


In this example, VXLAN tunnel is terminated. The egress is configured to send the packet out after adding VLAN tag 100.

- The VXLAN tunnel encapsulation includes IP, UDP and VXLAN header.
- The UDP destination port number in the UDP header is 4789, which indicates that the packet is a VXLAN encapsulated packet.

```
vtap(config-egress)# add egress-alias=egress-5 vlan=100
Egress Alias Configured successfully.
```

GRE tunnel termination - GRE tunnel initiation



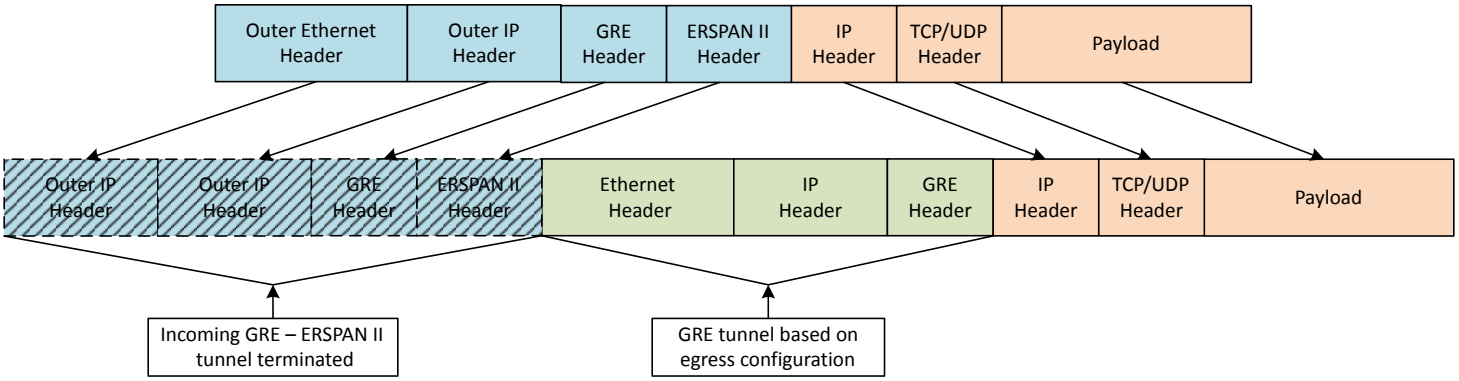
In this example, incoming GRE tunnel is terminated. The egress is configured to send the packet out in a GRE tunnel.

- When vTAP receives IP packet in a GRE tunnel, the outer IP header and GRE header are removed.
- The packet is routed based on egress configuration.

In this example, incoming GRE tunnel is terminated. The egress is configured to send the packet out in a GRE tunnel.

```
vtap(config-egress)# add egress-alias=egress-3 local-ip=2001::1 destination-ip=2001::2 nexthop-
mac=11:22:33:44:55:66 tunnel-type=gre
Egress Alias Configured successfully.
```

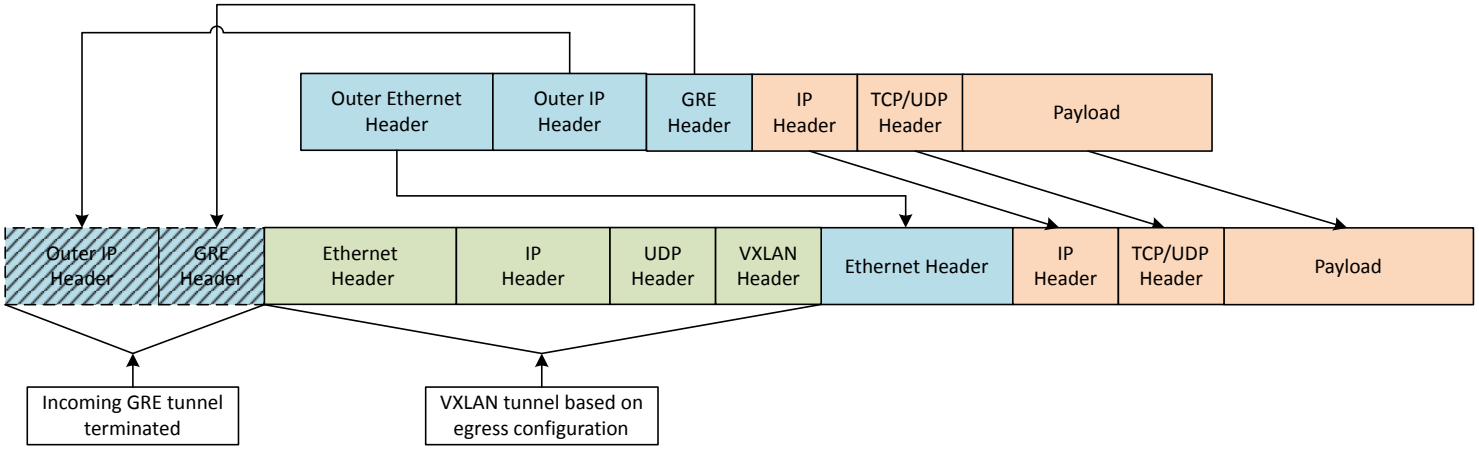
ERSPAN type II tunnel termination - GRE tunnel initiation



In this example, incoming ERSPAN type II tunnel is terminated. The egress is configured to send the packet out in a GRE tunnel.

```
vtap(config-egress)# add egress-alias=GRE_T1 local-ip=192.169.1.15 destination-ip=192.169.1.2 tunnel-
type=gre
Egress Alias Configured successfully.
```

GRE tunnel termination - VXLAN tunnel initiation



In this example, incoming GRE tunnel is terminated. The egress is configured to send the packet out in a VXLAN tunnel.

```
vtap(config-egress)# add egress-alias=vxlan_t1 local-ip=192.169.1.15 destination-ip=192.169.1.2 nexthop-
mac=00:0c:29:a1:b1:c3 vni=100
Egress Alias Configured successfully.
```

Purging stale flows

vTAP uses a periodic timeout-based cleanup process to purge stale flows.

The timeout interval starts 15 minutes after vTAP service starts and triggered every 15 minutes thereafter. Flows are monitored every 60 seconds in a fifteen minute cycle. During this period, if a flow does not receive packets for 60 seconds or if a TCP flow is received with FIN or RST packets, it is considered as stale and deleted.

vTAP takes 11 minutes to monitor and delete 500k stale flows, which is the maximum number of flows it supports.

> NOTE
> During stale cleanup, if all the flows in a tunnel are deleted, and if there are no other flows or tunnels associated with the parent tunnel, the tunnel is deleted.

Interface

An Interface in vTAP is a logical entity that represents the ingress port. vTAP includes a single interface with the traffic-type set to IP. The traffic-type cannot be changed.

This icon below is used throughout the document to represent the interface:

FIGURE 2 Interface icon

The characteristics and properties of an interface are:

- The interface logically maps to the Rx vNIC.
- The interface cannot be removed.
- One or more policies can be configured on the interface to be applied on the packets before they are sent out.

Configuring interface

TABLE 2 Interface parameters

Parameter	Description
mode	Specifies whether session and flow management is required by default. It can be set to either `packet` or `session`, which is the default mode. When the `mode` is set to `session`, the flow session management is enabled for any packet received with the supported protocols. When the `mode` is set to `packet`, all configured actions are applied on a per-packet basis, and flow session management is not enabled by default. If a SMARTMatch rule is configured with sampling or IPFIX, and the `mode` is set to `packet`, then flow sessions are maintained for the traffic on which such a rule is applied. If `mode` is modified from `session` to `packet` dynamically, then: • `sampling-policy` and `export-ipfix` properties of the interface are cleared dynamically. • All flow sessions are cleared except for the ones on which SMARTMatch rules with sampling or IPFIX actions are applied.
smatch-policy	Specifies the SMARTMatch policy to be applied on the interface before sending packets out. The SMARTMatch policy gets applied on both 'session' and 'packet'.
sampling-policy	Specifies the sampling policy to be applied on the interface before sending packets. The SMARTMatch policy gets applied only on 'session'.
egress	Specifies the egress path on which a flow/packet is to be sent out after processing. If the egress alias name is not specified, then it is set to `default-egress` and the packets are dropped post-processing.
export-ipfix	Specifies the option to export IPFIX for all: • Flows/packets sent to egress or • Dropped packets post-processing This parameter is applicable only on 'session'. It enables IPFIX metadata generation for flow sessions on which the SMARTMatch policy is not applied. When this parameter is set to `sampled-out`, the metadata is exported for the sampled-out flow sessions for which packets are being dropped by vTAP.
header-strip	Specifies the header to be stripped. One of the following values can be specified: • `802.1br_vntag`: For 802.1br and VN-tag stripping • `vxlan` • `nvgre` • `mpls` • `erspan` • `gtpu`
packet-slice	Specifies the offset value for packet slicing. A value between 1 and 1000 can be specified.

```
vtap> en
Password:
vtap# conf t

vtap(config)# interface
vtap(config-interface-ip)# set sampling-policy=sp1
Interface "ip"  parameters "Sampling policy sp1" updated successfully.
```

```
vtap(config-interface-ip)# set smatch-policy=smp1
Interface "ip"  parameters "Smartmatch policy smp1" updated successfully.

vtap(config-interface-ip)# set export-ipfix=all
Interface "ip"  parameters "export ipfix" updated successfully.

vtap(config-interface-ip)# set mode=session
Interface "ip"  parameters "mode" updated successfully.

vtap(config-interface-ip)# set header-strip=nvgre
Interface "ip"  parameters "header-strip" updated successfully.

vtap(config-interface-ip)# set packet-slice=10
Interface "ip"  parameters "packet-slice" updated successfully.

vtap(config-interface-ip)# set egress=gre_t1
Interface "ip"  parameters "egress gre_t1" updated successfully.


vtap(config-interface-ip)# sh
                     name  : ip
             traffic type  : ip
                   egress  : gre_t1
     sampling policy-name  : sp1
   smartmatch policy-name  : smp1
                     mode  : session
             export-ipfix  : all
             header-strip  : nvgre
             packet-slice  : 10
```

Packet modification

vTAP supports two types of packet modifications:

- Header stripping
- Packet slicing

Header stripping and packet slicing can be configured as an action on the interface or in a SMARTMatch rule. When both actions are configured, header stripping action is performed first, followed by packet slicing.

Header stripping

The header stripping feature checks packets for specified headers and tags, and removes them before sending the packets to the egress.

vTAP supports header stripping for:

- 802.1BR Tag
- VN-Tag
- VXLAN
- NVGRE
- MPLS label
- ERSPAN Type II
- GTP-U

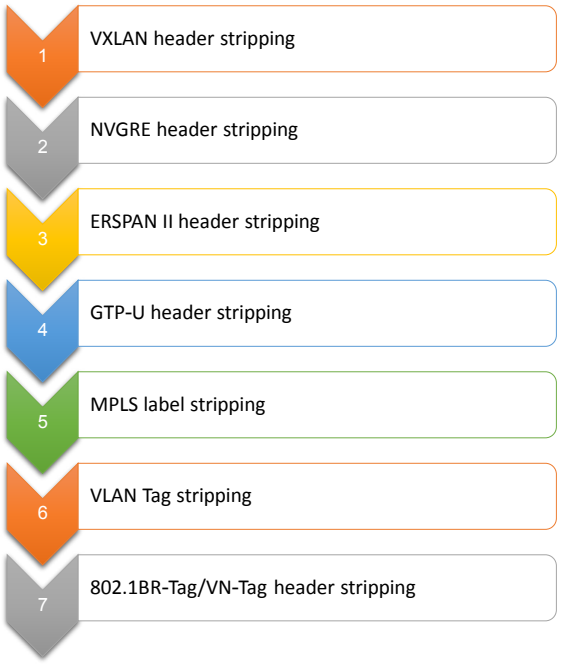
NOTE
If a packet matches a flex-match pattern, header stripping is performed on the packet only if there is no action configured for egress. If an action is configured for egress, the packet follows the egress path and the header stripping configuration is ignored.

Header stripping priority

Multiple header stripping configurations can be configured on the interface or in a SMARTMatch rule. However, only one header stripping operation is performed on a frame.

Following is the order in which the header stripping operation is performed:

FIGURE 3 Header stripping priority

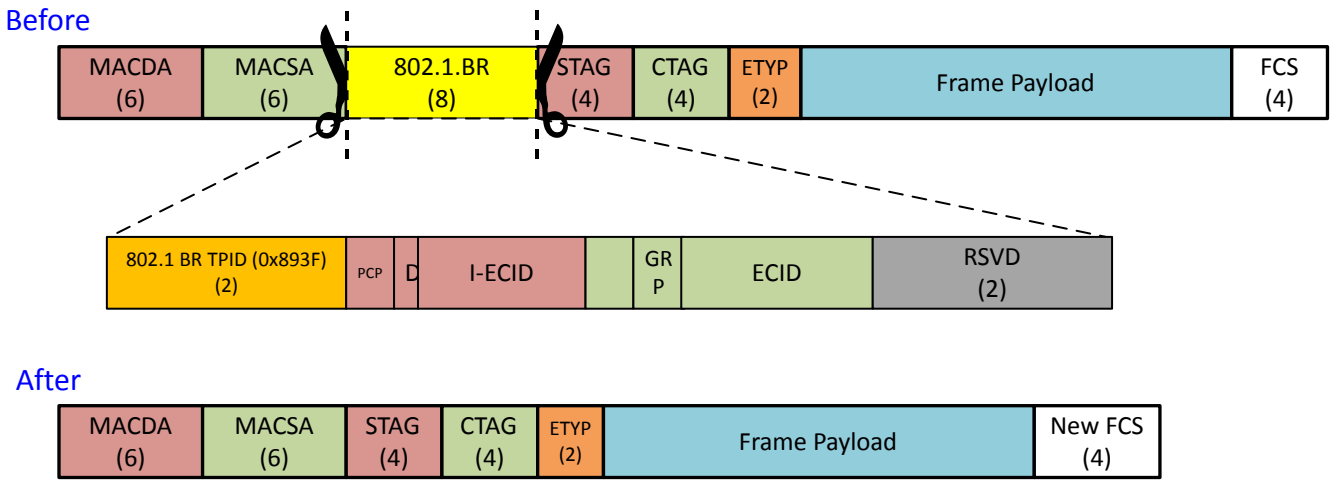


802.1BR tag

802.1BR tag is the first tag after MACSA and is 8 bytes long; this may be followed by S-Tag and/or C-Tag.

The following figure shows the structure and position of 802.1BR tag in a frame before and after 802.1BR tag stripping.

FIGURE 4 Before and after 802.1BR tag stripping



Configuring 802.1BR header stripping on the interface

- Run the `set header-strip=802.1br_vntag` command to set 802.1BR header stripping on the interface:

```
vtap> en
Password:
vtap# conf t

vtap(config)# interface
vtap(config-interface)# edit interface=ip
vtap(config-interface-ip)# set header-strip=802.1br_vntag
Interface "ip"  parameters "header-strip" updated successfully.

vtap(config-interface-ip)# sh
                    name  : ip
            traffic type  : ip
                  egress  : default-egress
     sampling policy-name  : sp1
   smartmatch policy-name  : smp1
                    mode  : session
            export-ipfix  : disabled
            header-strip  : 802.1br_vntag
            packet-slice  : -
```

- Run the `show interface-stats` command to verify:

```
vtap(config-interface-ip)# r
vtap(config-interface)# r
vtap(config)# r

vtap# show interface-stats
interface stats
    sampling policy-name : -

      smatch policy-name : -

      egress alias name  : drop
                 dropped packets : 159369
           header-strip : 802.1br_vntag
                        packets : 0
                          bytes : 0
           packet-slice : -
```

Configuring 802.1BR header stripping in a SMARTMatch rule

- Run the following command to set 802.1BR header stripping in a SMARTMatch rule:

```
vtap> en
Password:
vtap# conf t

vtap(config)# smartmatch
vtap(config-smartmatch)# edit smatch-policy=smp1
vtap(config-smartmatch-smp1)# add rule protocol=ip header-strip=802.1br_vntag

Rule with Rule ID 2 configured successfully for policy smp1

vtap(config-smartmatch-smp1)# sh

 smart-match policy 1
     name : smp1
    state : active
    rules :
          rule id : 1
          vlan id : 100
         protocol : tcp
         host1 ip : any
         host2 ip : any
```

```
      host1 port : any
      host2 port : any
     no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
     header-strip : -
     packet-slice : -

          rule id : 2
          vlan id : any
         protocol : ip
         host1 ip : any
         host2 ip : any
       host1 port : any
       host2 port : any
      no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
     header-strip : 802.1br_vntag
     packet-slice : -
```

- Run the `show smartmatch-stats rule rule-id=all` command to verify:

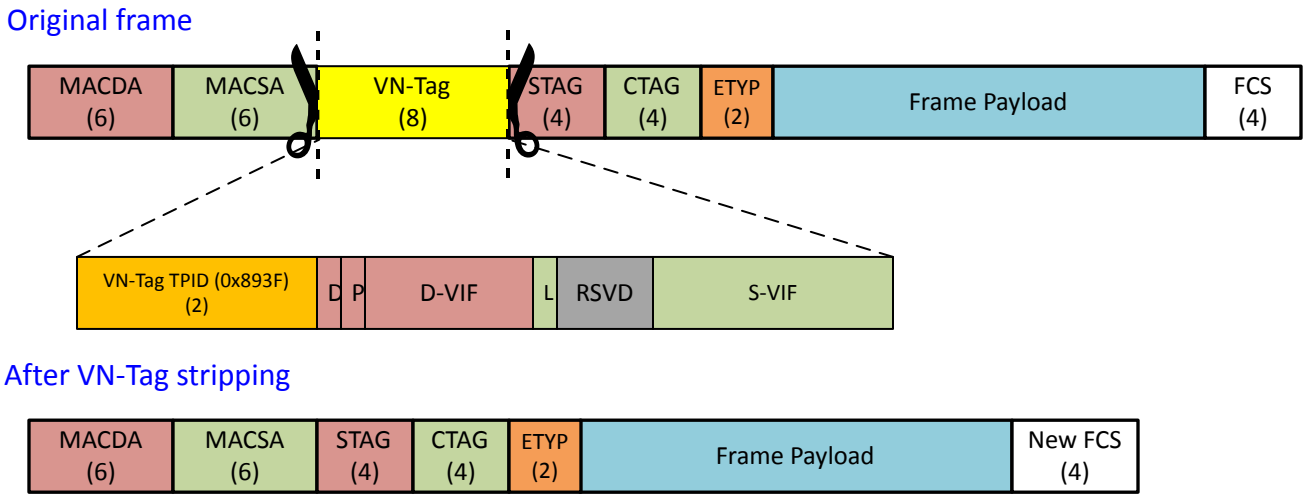
```
vtap(config-smartmatch-smp1)# r
vtap(config-smartmatch)# r
vtap(config)# r

vtap# show smartmatch-stats rule rule-id=all
  smatch policy name : smp1

 rule id : 1
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
     sampled packets : 0
    preserve packets : 0
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0

 rule id : 2
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
     sampled packets : 0
    preserve packets : 0
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0
```

VN-Tag

VN-Tag is the first tag after MACSA and is 6 bytes long; this may be followed by S-Tag and/or C-Tag.

The following figure shows the structure and position of VN-Tag in a frame before and after VN-Tag stripping.

FIGURE 5 Before and after VN-Tag stripping



Configuring VN-Tag header stripping on the interface

- Run the `set header-strip=802.1br_vntag` command to set VN-Tag header stripping on the interface:

```
vtap> en
Password:
vtap# conf t

vtap(config)# interface
vtap(config-interface)# edit interface=ip
vtap(config-interface-ip)# set header-strip=802.1br_vntag
Interface "ip"  parameters "header-strip" updated successfully.

vtap(config-interface-ip)# sh
                      name  : ip
              traffic type  : ip
                    egress  : default-egress
     sampling policy-name   : sp1
   smartmatch policy-name   : smp1
                      mode  : session
              export-ipfix  : disabled
              header-strip  : 802.1br_vntag
              packet-slice  : -
```

- Run the `show interface-stats` command to verify:

```
vtap(config-interface-ip)# r
vtap(config-interface)# r
vtap(config)# r

vtap# show interface-stats
interface stats
    sampling policy-name : -

      smatch policy-name : -

      egress alias name  : drop
                dropped packets : 159369
           header-strip : 802.1br_vntag
                        packets : 0
```

```
                    bytes : 0
          packet-slice : -
```

Configuring VN-Tag header stripping in a SMARTMatch rule

- Run the following command to set 802.1BR header stripping in a SMARTMatch rule:

```
vtap> en
Password:
vtap# conf t

vtap(config)# smartmatch
vtap(config-smartmatch)# edit smatch-policy=smp1
vtap(config-smartmatch-smp1)# add rule protocol=ip header-strip=802.1br_vntag

Rule with Rule ID 2 configured successfully for policy smp1

vtap(config-smartmatch-smp1)# sh

 smart-match policy 1
     name : smp1
    state : active
    rules :
          rule id : 1
          vlan id : 100
         protocol : tcp
         host1 ip : any
         host2 ip : any
       host1 port : any
       host2 port : any
      no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
     header-strip : -
     packet-slice : -

          rule id : 2
          vlan id : any
         protocol : ip
         host1 ip : any
         host2 ip : any
       host1 port : any
       host2 port : any
      no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
     header-strip : 802.1br_vntag
     packet-slice : -
```

- Run the `show smartmatch-stats rule rule-id=all` command to verify:

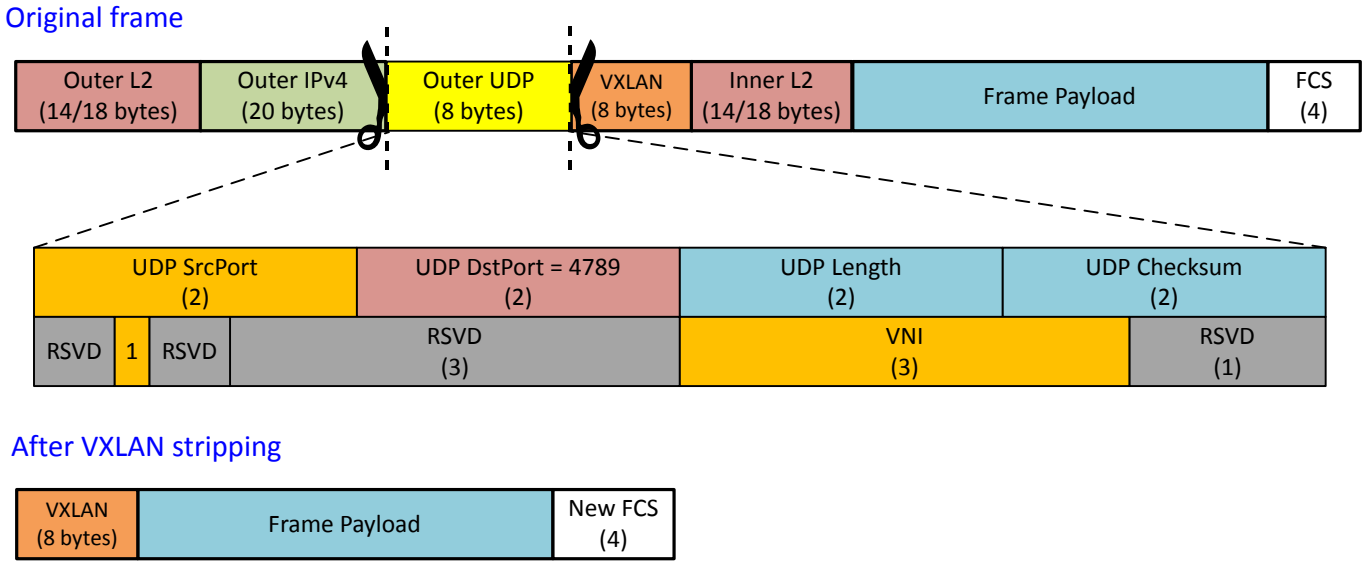
```
vtap(config-smartmatch-smp1)# r
vtap(config-smartmatch)# r
vtap(config)# r

vtap# show smartmatch-stats rule rule-id=all
  smatch policy name : smp1

 rule id : 1
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
```

```
     sampled packets : 0
    preserve packets : 0
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0

 rule id : 2
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
     sampled packets : 0
    preserve packets : 0
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0
```

VXLAN encapsulation

VXLAN encapsulated frames have an outer L2-IPv4-UDP-VXLAN header structure. To remove the encapsulation, these headers are discarded and the inner frame is exposed.

FIGURE 6 Before and after VXLAN stripping



Configuring VXLAN header stripping on the interface

- Run the `set header-strip=vxlan` command to set VXLAN header stripping on the interface:

```
vtap> en
Password:
vtap# conf t
```

```
vtap(config)# interface
vtap(config-interface)# edit interface=ip
vtap(config-interface-ip)# set header-strip=vxlan
Interface "ip"  parameters "header-strip" updated successfully.

vtap(config-interface-ip)# sh
                     name  : ip
             traffic type  : ip
                   egress  : default-egress
     sampling policy-name  : sp1
   smartmatch policy-name  : smp1
                     mode  : session
             export-ipfix  : disabled
             header-strip  : vxlan
             packet-slice  : -
```

- Run the `show interface-stats` command to verify:

```
vtap(config-interface-ip)# r
vtap(config-interface)# r
vtap(config)# r

vtap# show interface-stats
interface stats
    sampling policy-name : -

      smatch policy-name : -

      egress alias name  : drop
                 dropped packets : 159369
           header-strip : vxlan
                        packets : 0
                          bytes : 0
           packet-slice : -
```

Configuring VXLAN header stripping in a SMARTMatch rule

- Run the following command to set VXLAN header stripping in a SMARTMatch rule:

```
vtap> en
Password:
vtap# conf t

vtap(config)# smartmatch
vtap(config-smartmatch)# edit smatch-policy=smp1
vtap(config-smartmatch-smp1)# add rule protocol=ip header-strip=vxlan

Rule with Rule ID 2 configured successfully for policy smp1

vtap(config-smartmatch-smp1)# sh

 smart-match policy 1
     name : smp1
    state : active
    rules :
          rule id : 1
          vlan id : 100
         protocol : tcp
         host1 ip : any
         host2 ip : any
       host1 port : any
       host2 port : any
      no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
```

```
     header-strip : -
     packet-slice : -

          rule id : 2
          vlan id : any
         protocol : ip
         host1 ip : any
         host2 ip : any
       host1 port : any
       host2 port : any
      no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
     header-strip : vxlan
     packet-slice : -
```

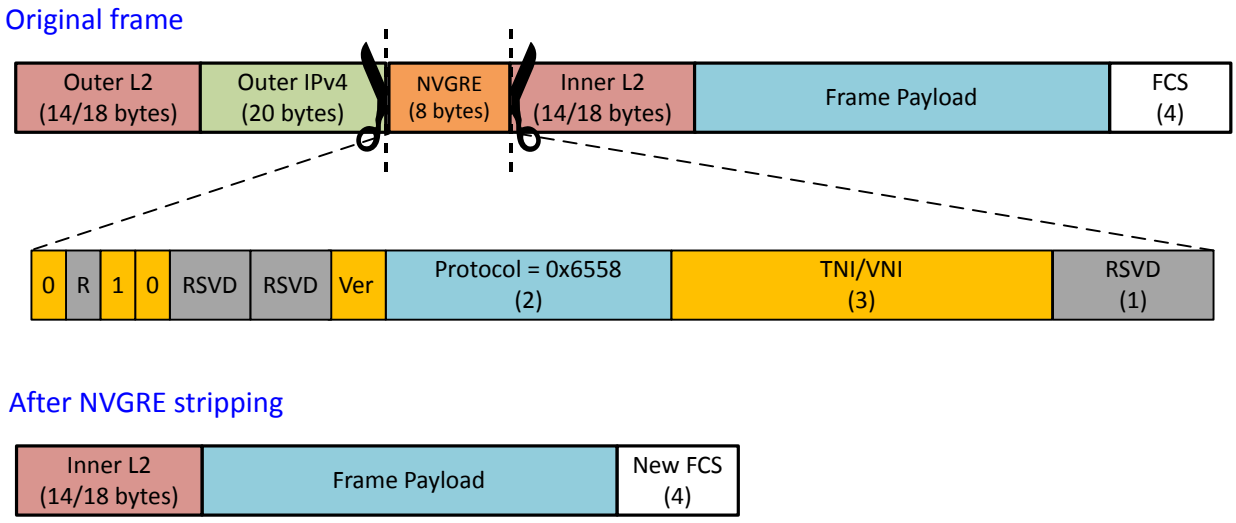
- Run the `show smartmatch-stats rule rule-id=all` command to verify:

```
vtap(config-smartmatch-smp1)# r
vtap(config-smartmatch)# r
vtap(config)# r

vtap# show smartmatch-stats rule rule-id=all
  smatch policy name : smp1

 rule id : 1
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
     sampled packets : 0
    preserve packets : 0
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0

 rule id : 2
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
     sampled packets : 0
    preserve packets : 0
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0
```

NVGRE encapsulation

NVGRE encapsulated frames have an outer L2-IPv4-NVGRE header structure. To remove the encapsulation, these headers are discarded and the inner frame is exposed.

FIGURE 7 Before and after NVGRE stripping



Configuring NVGRE header stripping on the interface

- Run the `set header-strip=nvgre` command to set NVGRE header stripping on the interface:

```
vtap> en
Password:
vtap# conf t

vtap(config)# interface
vtap(config-interface)# edit interface=ip
vtap(config-interface-ip)# set header-strip=nvgre
Interface "ip"  parameters "header-strip" updated successfully.

vtap(config-interface-ip)# sh
                      name  : ip
              traffic type  : ip
                    egress  : default-egress
     sampling policy-name   : sp1
   smartmatch policy-name   : smp1
                      mode  : session
              export-ipfix  : disabled
              header-strip  : nvgre
              packet-slice  : -
```

- Run the `show interface-stats` command to verify:

```
vtap(config-interface-ip)# r
vtap(config-interface)# r
vtap(config)# r
```

```
vtap# show interface-stats
interface stats
    sampling policy-name : -

      smatch policy-name : -

      egress alias name  : drop
                 dropped packets : 159369
            header-strip : nvgre
                         packets : 0
                           bytes : 0
            packet-slice : -
```

Configuring NVGRE header stripping in a SMARTMatch rule

- Run the following command to set NVGRE header stripping in a SMARTMatch rule:

```
vtap> en
Password:
vtap# conf t

vtap(config)# smartmatch
vtap(config-smartmatch)# edit smatch-policy=smp1
vtap(config-smartmatch-smp1)# add rule protocol=ip header-strip=nvgre

Rule with Rule ID 2 configured successfully for policy smp1

vtap(config-smartmatch-smp1)# sh

 smart-match policy 1
     name : smp1
    state : active
    rules :
          rule id : 1
          vlan id : 100
         protocol : tcp
         host1 ip : any
         host2 ip : any
       host1 port : any
       host2 port : any
      no of alias : 0
  sampling policy : --
     tunnel name : --
    export ipfix : disable
          egress : --
    header-strip : -
    packet-slice : -

          rule id : 2
          vlan id : any
         protocol : ip
         host1 ip : any
         host2 ip : any
       host1 port : any
       host2 port : any
      no of alias : 0
  sampling policy : --
     tunnel name : --
    export ipfix : disable
          egress : --
    header-strip : nvgre
    packet-slice : -
```

- Run the `show smartmatch-stats rule rule-id=all` command to verify:

```
vtap(config-smartmatch-smp1)# r
vtap(config-smartmatch)# r
vtap(config)# r
```

```
vtap# show smartmatch-stats rule rule-id=all
  smatch policy name : smp1

 rule id : 1
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
     sampled packets : 0
    preserve packets : 0
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0

 rule id : 2
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
     sampled packets : 0
    preserve packets : 0
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0
```

MPLS encapsulation

MPLS may encapsulate IP payload or entire Ethernet frame (pseudo wire). MPLS header stripping can discard up to 4 MPLS labels and if applicable, pseudo wire control word.

The following figure shows MPLS frame structure for IPoMPLS and ETHoMPLS, and the frame structure after MPLS label stripping.

FIGURE 8 Before and after MPLS - IP payload stripping

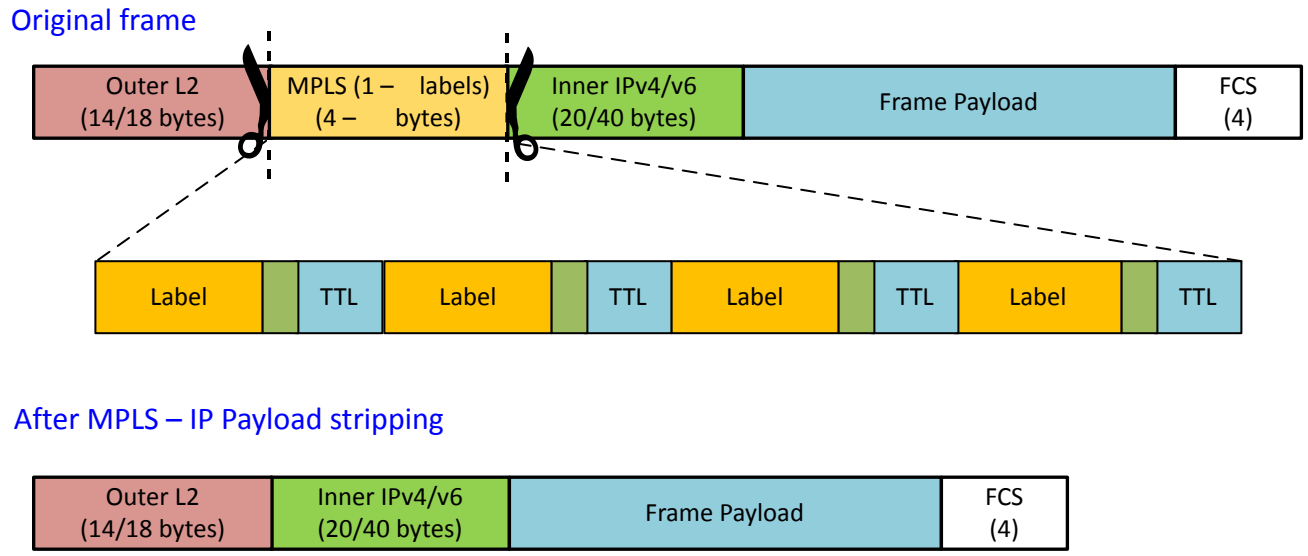
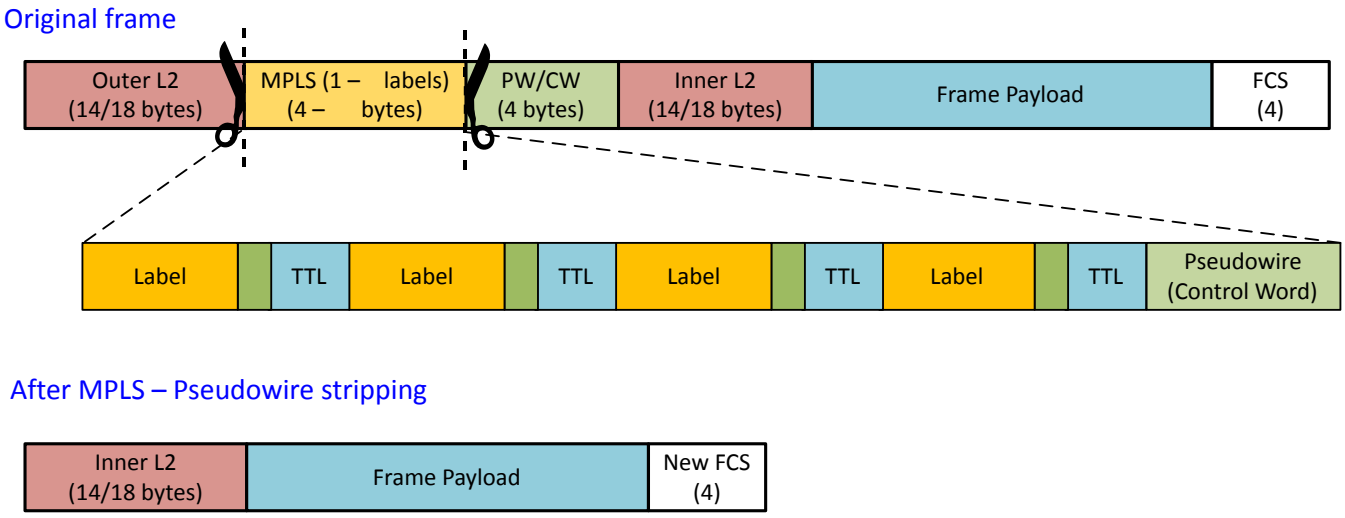


FIGURE 9 Before and after MPLS - Pseudowire stripping



Configuring MPLS header stripping on the interface

- Run the `set header-strip=mpls` command to set MPLS header stripping on the interface:

```
vtap> en
Password:
vtap# conf t
```

```
vtap(config)# interface
vtap(config-interface)# edit interface=ip
vtap(config-interface-ip)# set header-strip=mpls
Interface "ip"  parameters "header-strip" updated successfully.

vtap(config-interface-ip)# sh
                     name  : ip
             traffic type  : ip
                   egress  : default-egress
     sampling policy-name  : sp1
   smartmatch policy-name  : smp1
                     mode  : session
             export-ipfix  : disabled
             header-strip  : mpls
             packet-slice  : -
```

- Run the `show interface-stats` command to verify:

```
vtap(config-interface-ip)# r
vtap(config-interface)# r
vtap(config)# r

vtap# show interface-stats
interface stats
    sampling policy-name : -

      smatch policy-name : -

      egress alias name  : drop
                 dropped packets : 159369
           header-strip : mpls
                        packets : 0
                          bytes : 0
           packet-slice : -
```

Configuring MPLS header stripping in a SMARTMatch rule

- Run the following command to set MPLS header stripping in a SMARTMatch rule:

```
vtap> en
Password:
vtap# conf t

vtap(config)# smartmatch
vtap(config-smartmatch)# edit smatch-policy=smp1
vtap(config-smartmatch-smp1)# add rule protocol=ip header-strip=mpls

Rule with Rule ID 2 configured successfully for policy smp1

vtap(config-smartmatch-smp1)# sh

 smart-match policy 1
     name : smp1
    state : active
    rules :
          rule id : 1
          vlan id : 100
         protocol : tcp
         host1 ip : any
         host2 ip : any
       host1 port : any
       host2 port : any
      no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
     header-strip : -
```

```
      packet-slice : -

           rule id : 2
           vlan id : any
          protocol : ip
          host1 ip : any
          host2 ip : any
        host1 port : any
        host2 port : any
       no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
     header-strip : mpls
     packet-slice : -
```

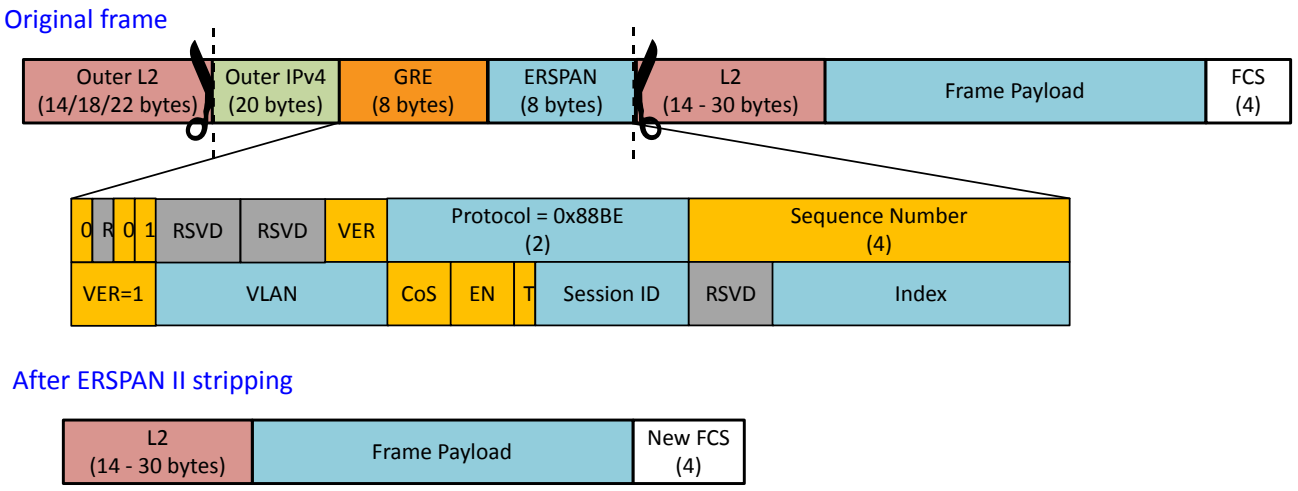
- Run the `show smartmatch-stats rule rule-id=all` command to verify:

```
vtap(config-smartmatch-smp1)# r
vtap(config-smartmatch)# r
vtap(config)# r

vtap# show smartmatch-stats rule rule-id=all
  smatch policy name : smp1

 rule id : 1
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
     sampled packets : 0
    preserve packets : 0
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0

 rule id : 2
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
     sampled packets : 0
    preserve packets : 0
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0
```

ERSPAN type II

ERSPAN-Type II has an outer L2-IPv4-GRE-ERSPAN header structure as shown in figure below. To remove the encapsulation, the outer headers are discarded, exposing the inner frame.

FIGURE 10 Before and after ERSPAN type II stripping



Configuring ERSPAN II header stripping on the interface

- Run the `set header-strip=erspan2` command to set ERSPAN II header stripping on the interface:

```
vtap> en
Password:
vtap# conf t

vtap(config)# interface
vtap(config-interface)# edit interface=ip
vtap(config-interface-ip)# set header-strip=erspan2
Interface "ip"  parameters "header-strip" updated successfully.

vtap(config-interface-ip)# sh
                      name  : ip
              traffic type  : ip
                    egress  : default-egress
     sampling policy-name   : sp1
   smartmatch policy-name   : smp1
                      mode  : session
              export-ipfix  : disabled
              header-strip  : erspan2
              packet-slice  : -
```

- Run the `show interface-stats` command to verify:

```
vtap(config-interface-ip)# r
vtap(config-interface)# r
vtap(config)# r

vtap# show interface-stats
interface stats
    sampling policy-name : -
```

```
      smatch policy-name : -

      egress alias name  : drop
                 dropped packets : 159369
            header-strip : erspan2
                         packets : 0
                           bytes : 0
            packet-slice : -
```

Configuring ERSPAN II header stripping in a SMARTMatch rule

- Run the following command to set ERSPAN II header stripping in a SMARTMatch rule:

```
vtap> en
Password:
vtap# conf t

vtap(config)# smartmatch
vtap(config-smartmatch)# edit smatch-policy=smp1
vtap(config-smartmatch-smp1)# add rule protocol=ip header-strip=erspan2

Rule with Rule ID 2 configured successfully for policy smp1

vtap(config-smartmatch-smp1)# sh

 smart-match policy 1
     name : smp1
    state : active
    rules :
          rule id : 1
          vlan id : 100
         protocol : tcp
         host1 ip : any
         host2 ip : any
       host1 port : any
       host2 port : any
      no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
     header-strip : -
     packet-slice : -

          rule id : 2
          vlan id : any
         protocol : ip
         host1 ip : any
         host2 ip : any
       host1 port : any
       host2 port : any
      no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
     header-strip : erspan2
     packet-slice : -
```

- Run the `show smartmatch-stats rule rule-id=all` command to verify:

```
vtap(config-smartmatch-smp1)# r
vtap(config-smartmatch)# r
vtap(config)# r

vtap# show smartmatch-stats rule rule-id=all
  smatch policy name : smp1
```

```
 rule id : 1
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
      dropped packets : 0
      sampled packets : 0
     preserve packets : 0
       number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0

 rule id : 2
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
      dropped packets : 0
      sampled packets : 0
     preserve packets : 0
       number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0
```

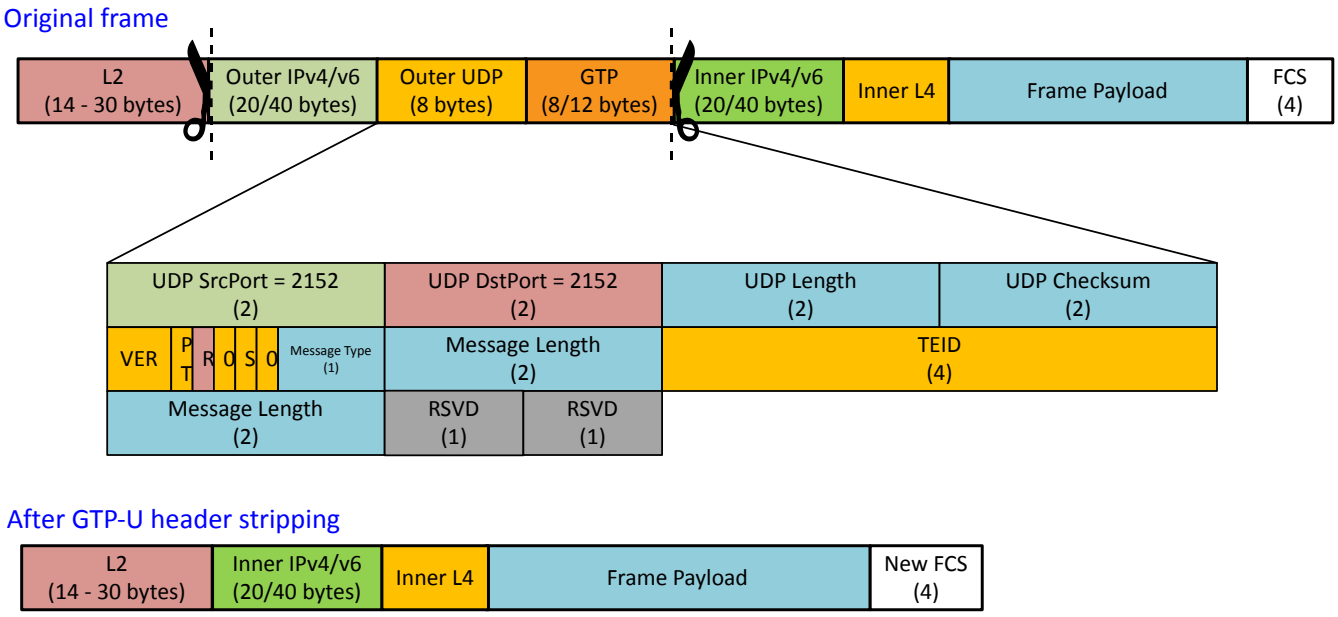
GTP-U

In a GTP-U-v1 encapsulated frame, the inner IPv4/v6 is tunneled in L2-IPv4/v6-UDP-GTP header as shown below. GTP-U header stripping removes the outer IP, UDP headers and GTP header, exposing the inner frame.

NOTE

vTAP supports GTP-U v1 with or without sequence number.

FIGURE 11 Before and after GTP-U header stripping



Configuring GTP-U header stripping on the interface

- Run the `set header-strip=gtpu` command to set GTP-U header stripping on the interface:

```
vtap> en
Password:
vtap# conf t

vtap(config)# interface
vtap(config-interface)# edit interface=ip
vtap(config-interface-ip)# set header-strip=gtpu
Interface "ip"  parameters "header-strip" updated successfully.

vtap(config-interface-ip)# sh
                      name  : ip
              traffic type  : ip
                    egress  : default-egress
     sampling policy-name   : sp1
   smartmatch policy-name   : smp1
                      mode  : session
              export-ipfix  : disabled
              header-strip  : gtpu
              packet-slice  : -
```

- Run the `show interface-stats` command to verify:

```
vtap(config-interface-ip)# r
vtap(config-interface)# r
vtap(config)# r

vtap# show interface-stats
interface stats
    sampling policy-name : -

      smatch policy-name : -
```

```
      egress alias name  : drop
                dropped packets : 159369
            header-strip : gtpu
                         packets : 0
                           bytes : 0
            packet-slice : -
```

Configuring GTP-U header stripping in a SMARTMatch rule

- Run the following command to set GTP-U header stripping in a SMARTMatch rule:

```
vtap> en
Password:
vtap# conf t

vtap(config)# smartmatch
vtap(config-smartmatch)# edit smatch-policy=smp1
vtap(config-smartmatch-smp1)# add rule protocol=ip header-strip=gtpu

Rule with Rule ID 2 configured successfully for policy smp1

vtap(config-smartmatch-smp1)# sh

 smart-match policy 1
     name : smp1
    state : active
    rules :
          rule id : 1
          vlan id : 100
         protocol : tcp
         host1 ip : any
         host2 ip : any
       host1 port : any
       host2 port : any
      no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
     header-strip : -
     packet-slice : -

          rule id : 2
          vlan id : any
         protocol : ip
         host1 ip : any
         host2 ip : any
       host1 port : any
       host2 port : any
      no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
     header-strip : gtpu
     packet-slice : -
```

- Run the `show smartmatch-stats rule rule-id=all` command to verify:

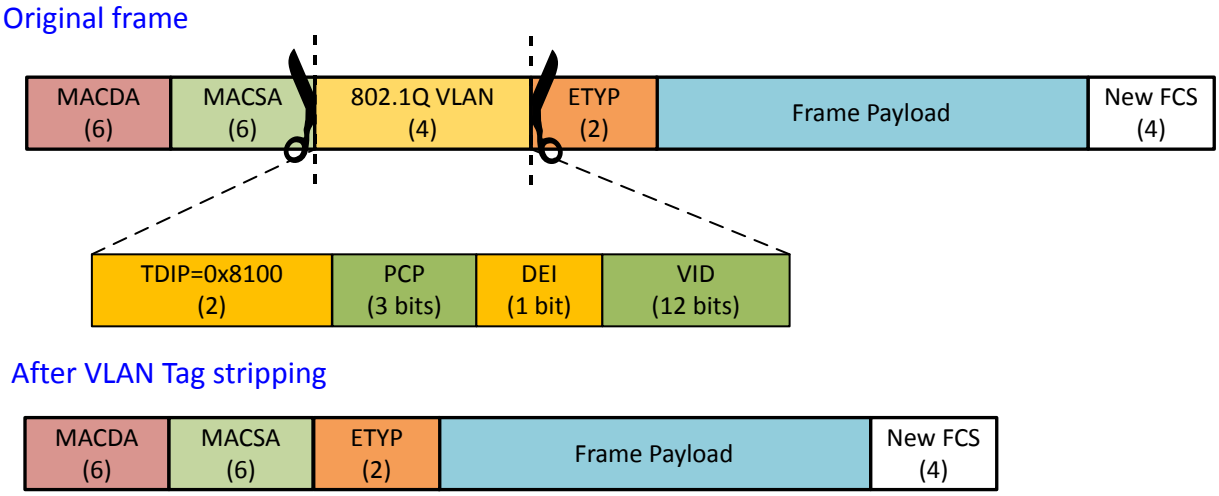
```
vtap(config-smartmatch-smp1)# r
vtap(config-smartmatch)# r
vtap(config)# r

vtap# show smartmatch-stats rule rule-id=all
  smatch policy name : smp1

 rule id : 1
```

```
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
     sampled packets : 0
    preserve packets : 0
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0

 rule id : 2
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
     sampled packets : 0
    preserve packets : 0
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0
```

VLAN Tag

802.1Q VLAN Tag is an optional 4-byte field in L2 ethernet frame. VLAN Tag stripping removes all the 802.1Q VLAN Tags from the packet.

FIGURE 12 Before and after VLAN Tag stripping



Configuring VLAN tag stripping on the interface

- Run the `set header-strip=vlan` command to set VLAN tag stripping on the interface:

```
vtap> en
Password:
vtap# conf t

vtap(config)# interface
vtap(config-interface)# edit interface=ip
vtap(config-interface-ip)# set header-strip=vlan
Interface "ip"  parameters "header-strip" updated successfully.

vtap(config-interface-ip)# sh
                     name  : ip
             traffic type  : ip
                   egress  : default-egress
     sampling policy-name  : sp1
   smartmatch policy-name  : smp1
                     mode  : session
             export-ipfix  : disabled
             header-strip  : vlan
             packet-slice  : -
```

- Run the `show interface-stats` command to verify:

```
vtap(config-interface-ip)# r
vtap(config-interface)# r
vtap(config)# r

vtap# show interface-stats
interface stats
    sampling policy-name : -

      smatch policy-name : -

      egress alias name  : drop
                 dropped packets : 159369
           header-strip : vlan
                        packets : 0
                          bytes : 0
           packet-slice : -
```

Configuring VLAN tag stripping in a SMARTMatch rule

- Run the following command to set VLAN tag stripping in a SMARTMatch rule:

```
vtap> en
Password:
vtap# conf t

vtap(config)# smartmatch
vtap(config-smartmatch)# edit smatch-policy=smp1
vtap(config-smartmatch-smp1)# add rule protocol=ip header-strip=vlan

Rule with Rule ID 2 configured successfully for policy smp1

vtap(config-smartmatch-smp1)# sh

 smart-match policy 1
     name : smp1
    state : active
    rules :
          rule id : 1
          vlan id : 100
         protocol : tcp
         host1 ip : any
         host2 ip : any
```

```
      host1 port : any
      host2 port : any
     no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
     header-strip : -
     packet-slice : -

          rule id : 2
          vlan id : any
         protocol : ip
         host1 ip : any
         host2 ip : any
       host1 port : any
       host2 port : any
      no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
     header-strip : vlan
     packet-slice : -
```

- Run the `show smartmatch-stats rule rule-id=all` command to verify:

```
vtap(config-smartmatch-smp1)# r
vtap(config-smartmatch)# r
vtap(config)# r

vtap# show smartmatch-stats rule rule-id=all
  smatch policy name : smp1

 rule id : 1
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
     sampled packets : 0
    preserve packets : 0
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0

 rule id : 2
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
     sampled packets : 0
    preserve packets : 0
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0
```

Packet slicing

Packet slicing is a feature that enables you to truncate incoming packets, while saving a specific number of bytes required for network analysis. In most cases, the first 120 bytes of a packet provides MAC, IP, TCP/UDP and other application data.

Packet slicing is performed based on the configured byte offset value. The packet is sliced after the specified offset.

When packet slicing is configured in a SMARTMatch rule, the offset value is calculated from the protocol. When it is configured on the interface, the offset value is calculated from the beginning of the packet.

NOTE

If a packet matches a flex-match pattern, it is sliced only if there is no action configured for egress. If an action is configured for egress, the packet follows the egress path and the packet is not sliced.

Configuring packet slicing on the interface

- Run the `set packet-slice=<offset>` command to set packet slicing offset value on the interface:

```
vtap> en
Password:
vtap# conf t

vtap(config)# interface
vtap(config-interface)# edit interface=ip
vtap(config-interface-ip)# set packet-slice=128
Interface "ip"  parameters "packet-slice" updated successfully.

vtap(config-interface-ip)# sh
                     name  : ip
             traffic type  : ip
                   egress  : default-egress
     sampling policy-name  : sp1
   smartmatch policy-name  : smp1
                     mode  : session
             export-ipfix  : disabled
             header-strip  : 802.1br_vntag
             packet-slice  : 400
```

- Run the `show interface-stats` command to verify:

```
vtap(config-interface-ip)# r
vtap(config-interface)# r
vtap(config)# r

vtap# show interface-stats
interface stats
    sampling policy-name : -

      smatch policy-name : -

      egress alias name  : drop
                 dropped packets : 159369
           header-strip : 802.1br_vntag
                        packets : 0
                          bytes : 0
           packet-slice : 400
                        packets : 3
                          bytes : 2116
```

Configuring packet slicing in a SMARTMatch rule

- Run the following command to set packet slicing offset value in a SMARTMatch rule:

```
vtap> en
Password:
vtap# conf t

vtap(config)# smartmatch
vtap(config-smartmatch)# edit smatch-policy=smp1
```

```
vtap(config-smartmatch-smp1)# add rule protocol=ip packet-slice=128

Rule with Rule ID 2 configured successfully for policy smp1

vtap(config-smartmatch-smp1)# sh

 smart-match policy 1
     name : smp1
    state : active
    rules :
          rule id : 1
          vlan id : 100
         protocol : tcp
         host1 ip : any
         host2 ip : any
       host1 port : any
       host2 port : any
      no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
     header-strip : -
     packet-slice : -

          rule id : 2
          vlan id : any
         protocol : ip
         host1 ip : any
         host2 ip : any
       host1 port : any
       host2 port : any
      no of alias : 0
  sampling policy : --
      tunnel name : --
     export ipfix : disable
           egress : --
     header-strip : 802.1br_vntag
     packet-slice : 128
```

- Run the `show smartmatch-stats rule rule-id=all` command to verify:

```
vtap(config-smartmatch-smp1)# r
vtap(config-smartmatch)# r
vtap(config)# r

vtap# show smartmatch-stats rule rule-id=all
  smatch policy name : smp1

 rule id : 1
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
     sampled packets : 0
    preserve packets : 0
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 0
  packet-slice bytes : 0

 rule id : 2
          rx packets : 0
            rx bytes : 0
          tx packets : 0
            tx bytes : 0
     dropped packets : 0
     sampled packets : 0
    preserve packets : 0
```

```
     number of flows : 0
header-strip packets : 0
  header-strip bytes : 0
packet-slice packets : 2
  packet-slice bytes : 720
```

SMARTMatch

The SMARTMatch feature enables vTAP to:

- Identify tunnels/flows/packets based on n-tuples
- Configure action for tunnels/flows/packets that match the n-tuples

A SMARTMatch rule also supports flex-match capability to detect one or more regex or hex patterns anywhere within the L2 packet boundary and configure action to either drop or forward the traffic to an egress path. If no action is configured for a flex-match, the respective counters are incremented for the detected pattern. If no forwarding or drop action is configured for the flex-match, the SMARTMatch rule actions are effective.

SMARTMatch does not always calculate the offset statically from the beginning of the packet. Based on the protocol, the offset is calculated from the end of protocol header. The following table specifies the protocols that SMARTMatch supports:

TABLE 3 SMARTMatch protocols and offset

Protocol	Description
Any	Offset starts from 0th Byte of Ethernet Frame.
Ether	Offset starts from 0th Byte of Ethernet Payload.
IP	Offset starts from 0th Byte of IP Payload (IPv4/IPv6).
TCP	Offset starts from 0th Byte of TCP Payload.
UDP	Offset starts from 0th Byte of UDP Payload.
SCTP	Offset starts from 0th Byte of SCTP Payload.
HTTP	Offset starts from 0th Byte of TCP Payload with TCP Port 80.
HTTPS	Offset starts from 0th Byte of TCP Payload with TCP Port 443.
SSH	Offset starts from 0th Byte of TCP Payload with TCP Port 22.

vTAP supports string, regex, and hex format for search pattern. The maximum search pattern length is 512 characters.

SMARTMatch policy

A SMARTMatch policy groups multiple generic packet match rules, each of which specify:

- Filter on n-tuple parameters
- Configure up to four flex-match aliases (optional)
- Configure a set of actions - forward to an egress destination or drop, sample flows, and export IPFIX metadata

Only one SMARTMatch policy can be configured in vTAP.

This icon below is used throughout the document to represent SMARTMatch policy:

FIGURE 13 SMARTMatch policy icon

Ingress tunnel

A SMARTMatch rule can be configured for a specific tunnel on the network. A tunnel name alias is used to identify an ingress tunnel on the network. While defining a tunnel alias, all parameters must be specified with an absolute value. The tunnel may or may not be terminated by vTAP.

A maximum of 100 ingress tunnels can be configured in vTAP.

Configuring ingress tunnel

An ingress tunnel can be VLAN, Q-in-Q, GRE, ERSPAN Type II, IPIP, or VXLAN.

> **NOTE**
> Nested tunnels cannot be configured. Therefore, GRE or IPIP or VXLAN cannot be configured along with VLAN or Q-in-Q.

TABLE 4 Ingress tunnel parameters

Parameter	Description
tunnel-name	Name of the ingress tunnel. The name is case-sensitive.
svlan	Specifies the Service VLAN (svlan) of the packet. **NOTE** `svlan` must be specified for VLAN.
cvlan	Specifies the Customer VLAN (cvlan) of the packet. **NOTE** Both `cvlan` and `svlan` must be specified for Q-in-Q.
host1-ip	Specifies the IP address of one host in the IP header.
host2-ip	Specifies the IP address of the other host in the IP header.
vni	Specifies VNI value for configuring VXLAN.
erspanII-vlan	VLAN Id for ERSPAN Type II tunnel. **NOTE** For ERSPAN Type II, specify `erspanII-vlan` with `tunnel-type` as `gre`.
tunnel-type	Enter one of the following values: • gre • vxlan

TABLE 4 Ingress tunnel parameters (continued)

Parameter	Description
	• ipip

TABLE 5 Ingress configuration

Ingress	Configuration
VXLAN	Specify VNI value for configuring VXLAN. **NOTE** The destination port for VXLAN is 4789. This value is fixed and cannot be changed. **Example** `add tunnel-name=t5 host1-ip=10.0.0.1 host2-ip=10.0.0.2 vni=100`
ERSPAN Type II	Protocol as GRE and ERSPAN VLAN is mandatory. **Example** `add tunnel-name=t6 host1-ip=2001::1 host2-ip=2002::2 erspanIIvlan=100 tunnel-type=gre`

VLAN or Q-in-Q	For VLAN, `svlan` must be used.
	Example - VLAN
	`add tunnel-name=t3 svlan=100`
	For Q-in-Q, `svlan` and `cvlan` must be used.
	Example - Q-in-Q
	`add tunnel-name=t4 svlan=120 clan=1055`

IPIP	**Example - IPIP** `add tunnel-name=t4 host1-ip=2001::1 host2-ip=2002::2 tunnel-type=ipip`
GRE	**Example - GRE** `add tunnel-name=t3 host1-ip=2001::1 host2-ip=2002::2 tunnel-type=gre`

Example

```
vtap> en
Password:
vtap# conf t

vtap(config)# smartmatch

vtap(config-smartmatch)# intunnel

vtap(config-smartmatch-intunnel)# add ?
tunnel-name=<tunnel-name> svlan=<svlan-id>
tunnel-name=<tunnel-name> svlan=<svlan-id> cvlan=<cvlan-id>
tunnel-name=<tunnel-name> host1-ip=<host1-ip-address> host2-ip=<host2-ip-address> tunnel-type=<GRE>
tunnel-name=<tunnel-name> host1-ip=<host1-ip-address> host2-ip=<host2-ip-address> erspanII-
vlan=<vlan-id> tunnel-type=<GRE>
tunnel-name=<tunnel-name> host1-ip=<host1-ip-address> host2-ip=<host2-ip-address> tunnel-
type=<IPIP>
tunnel-name=<tunnel-name> host1-ip=<host1-ip-address> host2-ip=<host2-ip-address> vni=<vxlan-vni>

vtap(config-smartmatch-intunnel)#add tunnel-name=t1 svlan=100
```

```
Ingress tunnel configured successfully.

vtap(config-smartmatch-intunnel)#add tunnel-name=t2 svlan=120 cvlan=1055
Ingress tunnel configured successfully.

vtap(config-smartmatch-intunnel)# add tunnel-name=t3 host1-ip=2001::1 host2-ip=2002::2 tunnel-
type=gre
Ingress tunnel configured successfully.

vtap(config-smartmatch-intunnel)# add tunnel-name=t4 host1-ip=2001::1 host2-ip=2002::2 tunnel-
type=ipip
Ingress tunnel configured successfully.

vtap(config-smartmatch-intunnel) add tunnel-name=t5 host1-ip=10.0.0.1 host2-ip=10.0.0.2 vni=100
Ingress tunnel configured successfully.

vtap(config-smartmatch-intunnel)# add tunnel-name=t6 host1-ip=2001::1 host2-ip=2002::2
erspanIIvlan=100 tunnel-type=gre
Ingress tunnel configured successfully.
```

Run the `show tunnel-name= [ all | <ingress-tunnel-name> ]` to display the configured ingress tunnels.

```
vtap(config-smartmatch-intunnel)# show tunnel-name=all

              tunnel-name  : t5
                 host1-ip  : 10.0.0.1
                 host2-ip  : 10.0.0.2
         destination-port  : 4789
              tunnel-type  : vxlan
                      vni  : 100

              tunnel-name  : t2
                    svlan  : 120
                    cvlan  : 1055

              tunnel-name  : t4
                 host1-ip  : 2001::1
                 host2-ip  : 2002::2
              tunnel-type  : ipip

              tunnel-name  : t1
                     vlan  : 100

              tunnel-name  : t6
                 host1-ip  : 2001::1
                 host2-ip  : 2002::2
            erspanII-vlan  : 100
              tunnel-type  : gre-erspanII

              tunnel-name  : t3
                 host1-ip  : 2001::1
                 host2-ip  : 2002::2
              tunnel-type  : gre
```

Pattern matching

A flex-match pattern can be configured by creating an alias, giving it a name and adding the pattern. An alias can include up to four flex-match patterns.

vTAP supports regex and hex format for the search pattern. PCRE is the supported format for regex. The maximum search pattern length is 512 characters.

NOTE
In the case of multiple tunnels, if the protocol is IP, flex-match is applied only on the payload that follows the outer tunnel (which is specified in the rule). However, if the specified tunnel follows VLAN/Q-in-Q, flex-match is applied on the payload of the VLAN/Q-in-Q.

TABLE 6 Alias configuration

Parameter	Description
alias	Specifies the name of the alias.
flex-match	Specifies the regex/hex pattern at a specific offset. The maximum search pattern length is 512 characters. NOTE Flex-match pattern cannot include the following characters: Space, pipe (**|**), colon (**:**) and comma (**,**). NOTE All strings must be added between two backtick symbols (`). **Example** • Adding string **abc*.com**: `add alias=a1 flex-match=0:0:`abc*.com`` • Adding string **extremenet**: `add alias=a1 flex-match=0:0:`extremenet`` • Adding **?** for string matching: `add alias=a1 flex-match=0:0:`(?i)pattern``

This flex-match `10:10:pattern1` indicates that the start offset is `10` and the end offset is `20` (start offset + length = 10 + 10 = 20) for the search pattern `pattern1`. The offset values are calculated based on the n-tuple match that uses it.

Example 1

```
vtap> en
Password:
vtap# conf t


vtap(config)# smartmatch
vtap(config-smartmatch)# alias
vtap(config-smartmatch-alias)# add alias=alias1 flex-match=10:10:`pattern1`,20:20:`pattern2`,
30:30:`pattern3`,40:40:`pattern4`
Match criteria updated successfully.

vtap(config-smartmatch-alias)# show alias=all

 Flex match combination 1
        alias name                    : alias1
        Number of flex matches        : 4
          Flex match1 byte offset     : 10
          Flex match1 length to search: 10
          Flex match1 pattern         : pattern1
          Flex match2 byte offset     : 20
          Flex match2 length to search: 20
          Flex match2 pattern         : pattern2
          Flex match3 byte offset     : 30
          Flex match3 length to search: 30
          Flex match3 pattern         : pattern3
          Flex match4 byte offset     : 40
          Flex match4 length to search: 40
          Flex match4 pattern         : pattern4
```

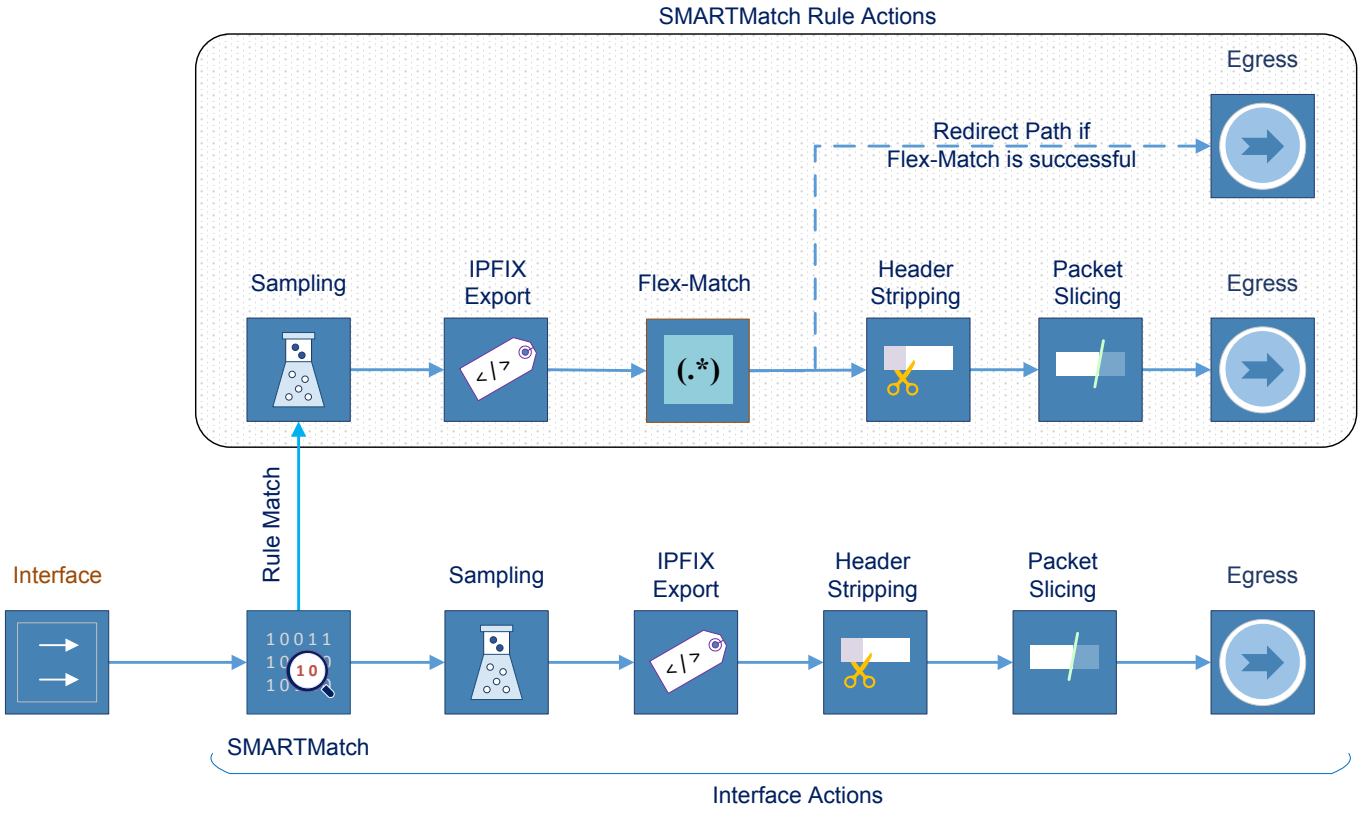
SMARTMatch rule

A SMARTMatch rule must have at least one match parameter.

> **NOTE**
> A SMARTMatch rule is applied on a Flow if its source/destination IP address/Ports match the rule parameters. The rules are applicable on both directions of a Flow.

The image below shows a SMARTMatch rule, the supported actions and the order in which these actions are executed:

FIGURE 14 SMARTMatch rule actions



SMARTMatch rules do not have any set priority. Instead the rules are selected based on 'Longest Prefix Match' criteria. When a new rule with the longest prefix match is added, it overrides all the other rules. However, this does not result in re-evaluation of existing rules applied across the system.

For example:

- There are ten flows with IP address of one endpoint in the range 10.10.0.0/16 and five of these flows have the other endpoint's IP address in the range 20.20.20.0/24.
- The first rule (**Rule 1**) is added with a sampling policy `sp1` with 50% sampling:

  Rule 1: `add rule host1-ip=10.10.0.0/16 sampling-policy=sp1 egress=e1`. The rule match has 1 tuple.

This rule samples-out five flows with source IP address in the range 10.10.0.0/16.

- Assume that because of rule 1, each of the five flows that were sampled out had the other endpoint IP address in the range 20.20.20.0/24.
- Next, another rule (**Rule 2**) is added to the same SMARTMatch policy:

  Rule 2: `add rule host1-ip=10.10.0.0/16 host2-ip=20.20.20.0/24 egress=e2`. The rule match has 2 tuples.
- Since Rule 2 has two tuples and is longer than Rule 1, this rule matches the five flows with the endpoint IP addresses in the range 10.10.0.0/16 and 20.20.20.0/24 respectively. This modifies the action for these five flows immediately, and forwards the packets of these flows to the egress `e2`, which were previously sampled out due to Rule 1.
- Addition of Rule 2 does not result in re-evaluation of Rule 1 and Rule 1 is not executed to fulfill its 50% sampling configuration due to the impact of Rule 2. However, if Rule 1 is deleted and added again, it is then considered to be a new rule and executed accordingly.

Configuring SMARTMatch policy and rule

TABLE 7 SMARTMatch rule configuration

Parameter	Description
protocol	This is the protocol of the IP traffic. It may also specify the protocol within a particular type of payload. The supported values are `any`, `ether`, `ip`, `tcp`, `udp`, `https`, `http`, `ssh`, and `sctp`. **NOTE** The protocol values `none`, `ether` and `ip` are used only for offset calculation and matches with any protocol received in the packet.
vlan	Specifies the VLAN of the packet.
host1-ip	Specifies the IP address of one host in the IP header.
host2-ip	Specifies the IP address of the other host in the IP header.
host1-port	Specifies the port used by host 1.
host2-port	Specifies the port used by host 2.
smatch-alias	Specifies the name of the SMARTMatch alias to be used in the rule.
tunnel-name	Specifies the name of the ingress tunnel to be used in the rule. • When a rule specifies a tunnel-name with flow parameters, the tunnel is interpreted as the immediate parent tunnel of the flow/tunnel (and not a grand-parent in the hierarchy). Hierarchical tunnels are not supported. • A SMARTMatch rule with an ingress tunnel tunnel-name and flow parameters, the tunnel is interpreted as the immediate parent tunnel of the flow/tunnel (and not a grand-parent in the hierarchy). A SMARTMatch rule does NOT support configuration of hierarchical tunnels in this release. • If a SMARTMatch rule specifies an ingress tunnel (`tunnel-name`) of type VLAN or Q-in-Q, the flow tuples cannot contain VLAN with an absolute value as a parameter in the match criteria. **Example** `config-smartmatch-p1#add rule tunnel-name=t1 vlan=10` Not allowed if `t1` is Q-in-Q or vlan • A SMARTMatch rule without an ingress tunnel (`tunnel-name`) is applied on all the flows that are not inside a tunnel
egress	Specifies the egress path on which a flow/packet is to be sent out after processing. The egress may be set to `drop` to drop packets post-processing.
sampling-policy	Specifies the name of the sampling policy.
export-ipfix	Set the option to export IPFIX for all: • Flows/packets sent to egress or

TABLE 7 SMARTMatch rule configuration (continued)

Parameter	Description
	• Dropped packets post-processing This parameter is applicable only on 'session'. It enables IPFIX metadata generation for flow sessions on which the SMARTMatch policy is not applied. When this parameter is set to `sampled-out`, the metadata is exported for the sampled-out flow sessions for which packets are being dropped by vTAP.
packet-slice	Specifies the offset value for packet slicing. A value between 1 and 1000 can be specified.
header-strip	Specifies the header to be stripped. One of the following values can be specified: • `802.1br_vntag` • `vxlan` • `nvgre` • `mpls` • `erspan` • `gtpu`

Example

```
vtap> en
Password:
vtap# conf t

vtap(config)# smartmatch
vtap(config-smartmatch)# add smatch-policy=smp1
vtap(config-smartmatch-smp1)# add rule vlan=400 protocol=tcp host1-ip=10.0.0.1 host2-ip=10.0.0.2 smatch-
alias=alias1:egress-1,alias2:egress-2,alias3:drop,alias4:drop sampling-policy=sampling-1 export-
ipfix=sampled-out header-strip=vxlan,nvgre packet-slice=72
Rule with Rule ID 1 configured successfully for policy smp1

vtap(config-smartmatch-smp1)#add rule vlan=any host1-ip=10.0.0.1 host2-ip=100.1.1.1 egress=drop export-
ipfix=all
Rule with Rule ID 2 configured successfully for policy smp1

vtap(config-smartmatch-smp1)#add rule vlan=any host1-ip=2001:43:12::1 host2-ip=2600:32:11::1
egress=egress-2 export-ipfix=disable
Rule with Rule ID 3 configured successfully for policy smp1

vtap(config-smartmatch-smp1)# show smatch-policy=all
smart-match policy 1
    name : smp1
   state : inactive
   rules :
         rule id : 1
         vlan id : any
        protocol : any
        host1 ip : 10.0.0.1
        host2 ip : 100.1.1.1
      host1 port : any
      host2 port : any
     no of alias : 0
 sampling policy : --
     tunnel name : --
    export ipfix : all
          egress : drop
    header-strip : -
    packet-slice : -

         rule id : 2
         vlan id : 400
        protocol : tcp
        host1 ip : 10.0.0.1
        host2 ip : 10.0.0.2
      host1 port : any
```

```
     host2 port : any
    no of alias : 1
          alias name 1 : a1
sampling policy : sp1
    tunnel name : --
   export ipfix : sampled-out
         egress : --
   header-strip : vxlan,nvgre
   packet-slice : 72

        rule id : 3
        vlan id : any
       protocol : any
       host1 ip : 2001:43:12::1
       host2 ip : 2600:32:11::1
     host1 port : any
     host2 port : any
    no of alias : 0
sampling policy : --
    tunnel name : --
   export ipfix : disable
         egress : egress-1
   header-strip : -
   packet-slice : -
```

Sampling

Sampling provides a representative view of IP traffic. Sampling in vTAP can be achieved by configuring a sampling policy.

Sampling policy

Sampling policies are applicable only on flows, and not packets.

Sampling policies have no rules or configured match criteria. A maximum of 10 sampling policies can be configured in vTAP. A sampling policy can be used as an Action in a SMARTMatch Rule.

NOTE

- A sampling policy cannot be deleted if it is already applied on the interface or in a SMARTMatch rule. To delete a sampling policy, remove it from the interface and all the SMARTMatch rules and delete the policy.
- If a SMARTMatch policy and a sampling policy are applied on the interface, the SMARTMatch policy has higher priority.

This icon below is used throughout the document to represent sampling policy:

FIGURE 15 Sampling policy icon

Configuring sampling policy

TABLE 8 Sampling policy parameters

Parameter	Description
sample-rate	Specifies the sampling drop percentage for the flow. Enter a value between 1 and 100. The default value is 1. NOTE If the sampling rate for a sampling policy is cleared (using `clear sample-rate`), the sampling rate for that policy is set to 1.
preserve-pkts	When a flow is selected to be sampled out, this parameter specifies the number of packets that must be forwarded for that flow before it is dropped. If this parameter is not set, the default value is set to `0` and the flow is dropped immediately. Enter a value between 1 and 1000. The default value is 0.

```
vtap> en
Password:
vtap# conf t

vtap(config)# sampling
vtap(config-sampling)# add sampling-policy=sp1

vtap(config-sampling-sp1)# set sample-rate=20
sampling policy "sp1" parameter "sampling " configured successfully.
```

```
vtap(config-sampling-sp1)# set preserve-pkts=100
sampling policy "sp1" parameter "preserve-pkts" configured successfully.

vtap(config-sampling-sp1)# show sampling-policy=all
policy name : sp1
           state : inactive
     sample-rate : 10
   preserve-pkts : 100

vtap(config-sampling-sp1)# clear sample-rate
sampling policy "sp1" with parameter "sample-rate" cleared successfully.

vtap(config-sampling-sp1)# sh
policy name : sp1
           state : inactive
     sample-rate : 1
   preserve-pkts : 10
```

IPFIX flow exporter

vTAP supports IP Flow Information Export (IPFIX), which is based on the Internet Engineering Task Force (IETF) standard. IPFIX is used to collect IP Flow information from Switches, Routers and other network devices, and analyze the Traffic Flow information. UDP is the supported transport protocol.

vTAP includes a dedicated virtual interface for IPFIX export. Alternately, the management vNIC can also be used for IPFIX export.

NOTE
Do not use Rx, Tx or Test vNIC for exporting IPFIX.

vTAP supports two IPFIX collectors. When two collectors are configured, the IPFIX packets are broadcast to both collectors.

Data templates for the system and flows

vTAP supports IPFIX templates for:

- vTAP node
- Tunnel flows
- IP flow sessions
- DNS application
- SSL/TLS certificates
- HTTP

The following is the IE Type Options Template for the Extreme custom IEs:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Set ID = 3           |          Length = 42          | 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Template ID = 300       |        Field Count = 8        | 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Scope Field Count = 2     |0|      PEN = 346              | 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Field Length = 4        |0|   IE ID = 303               | 4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Field Length = 2        |0|   IEDataType = 339          | 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Field Length = 1        |0|   IESemantics = 344         | 6
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Field Length = 1        |0|   IERangeBegin = 342        | 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Field Length = 8        |0|   IERangeEnd = 343          | 8
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Field Length = 8        |0|   IEName = 341              | 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Field Length = 65535      |0|   IEDescription = 340       | 10
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Field Length = 65535      |                               11
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Element Type Records for each specific IEs is formed as per the following format:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Set ID = 300         |        Length = <length>      | 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  PEN = Extreme Networks (1916)                | 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         ID = <hex value>      |   DT= <hex>   | DT Sem.=<hex> | 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Range Begin = <hex value>               | 4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Range End = <hex value>                | 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Length = <hex>|              Name = <IE Name>                 | 6
+-+-+-+-+-+-+-+-+                                              +
|                                                               | ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Length = <hex>|    Description = <IE Description>            | 6+n
+-+-+-+-+-+-+-+-+                                              +
|                                                               | ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TABLE 9 Template for vTAP node

Element ID	Name	Data Type	Data Type Semantics	Description
85	octetTotalCount	Unsigned64	totalCounter	The total number of octets in incoming packets for this Flow at the Observation Point since the Metering Process (re-)initialization for this Observation Point. The number of octets includes IP header(s) and IP Payload.
86	packetTotalCount	unsigned64	totalCounter	The total number of incoming packets for this Flow at the Observation Point since the Metering Process (re-)initialization for this Observation Point.
135	droppedPacketTotalCount	unsigned64	totalCounter	The number of packets for this Flow dropped by packet treatment since the Metering Process (re-)initialization for this Observation Point.
134	droppedOctetTotalCount	unsigned64	totalCounter	The total number of octets in packets for this Flow dropped by packet treatment since the Metering Process (re-)initialization for this Observation Point. The number of octets includes IP header(s) and IP payload.
3001	observedTunnelTotalCount	unsigned64	totalCounter	The total number of Tunnels present in vTAP.
3002	terminatedTunnelTotalCount	unsigned64	totalCounter	The total number of Tunnels terminated by vTAP.
163	observedFlowTotalCount	unsigned64	totalCounter	The total number of Flows observed in the

TABLE 9 Template for vTAP node (continued)

Element ID	Name	Data Type	Data Type Semantics	Description
				Observation Domain since the Metering Process (re-)initialization for this Observation Point. This indicates the number of flows in vTAP.
166	notSentFlowTotalCount	unsigned64	totalCounter	The total number of Flow Records that were generated by the Metering Process and dropped by the Metering Process or by the Exporting Process instead of being sent to the Collecting Process.

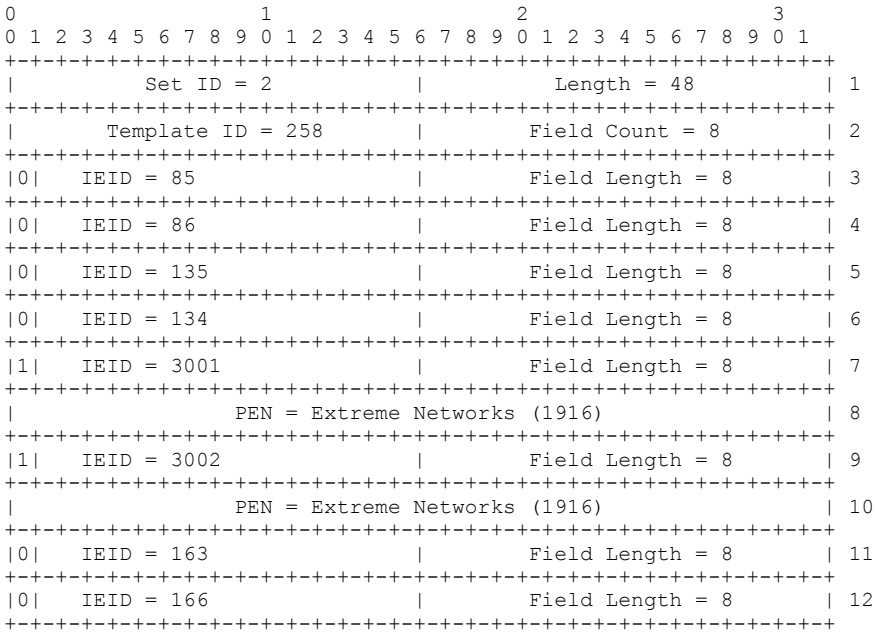


TABLE 10 Template for tunnel flows

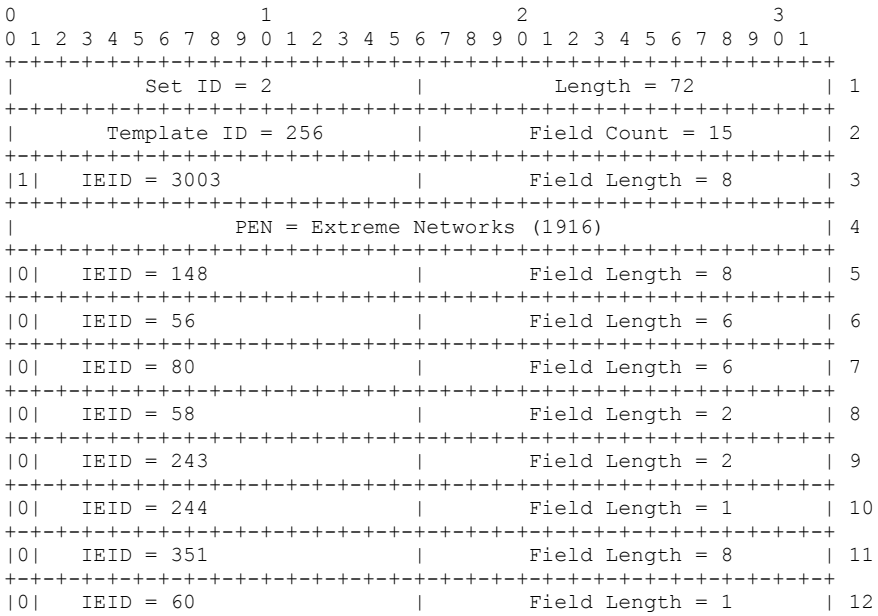
Element ID	Name	Data Type	Data Type Semantics	Description
3003	parentTunnelId	unsigned64	Identifier	The Tunnel ID that identifies the parent of this tunnel in vTAP.
148	flowId	Unsigned64	Identifier	An identifier of a Flow that is unique within an Observation Domain. This Information Element can be used to distinguish between different Flows if Flow Keys such as IP addresses and port numbers are not reported or are reported in separate records.

TABLE 10 Template for tunnel flows (continued)

Element ID	Name	Data Type	Data Type Semantics	Description
56	sourceMacAddress	macAddress	default	The IEEE 802 source MAC address field. This indicates the MAC address of one endpoint.
80	destinationMacAddress	macAddress	default	The IEEE 802 destination MAC address field. This indicates the MAC address of the other endpoint.
58	vlanId	unsigned16	identifier	Virtual LAN identifier associated with ingress interface.
243	dot1qVlanId	unsigned16	identifier	The value of the 12-bit VLAN Identifier portion of the Tag Control Information field of an Ethernet frame.
244	dot1qPriority	unsigned8	identifier	The value of the 3-bit User priority portion of the Tag Control Information field of an Ethernet frame
351	layer2SegmentId	unsigned64	identifier	Identifier of a layer 2 network segment in an overlay network. The vTAP will include the VNI information in this field.
296	grekey	unsigned32	Identifier	GRE key used for identifying an individual traffic flow within a tunnel.
60	ipVersion	unsigned8	identifier	The IP version field in the IP packet header.
8	sourceIPv4Address	ipv4Address	default	The IPv4 source address in the IP packet header. This indicates the IPv4 address of one endpoint.
12	destinationIPv4Address	ipv4Address	default	The IPv4 destination address in the IP packet header. This indicates the IPv4 address of the other endpoint.
27	sourceIPv6Address	ipv6Address	default	The IPv6 source address in the IP packet header. This indicates the IPv6 address of one endpoint.
28	destinationIPv6Address	ipv6Address	default	The IPv6 destination address in the IP packet header. This indicates the IPv6 address of the other endpoint.
163	observedFlowTotalCount	unsigned64	totalCounter	The total number of Flows observed in the Observation Domain since the Metering Process (re-)initialization for this Observation Point.

TABLE 10 Template for tunnel flows (continued)

Element ID	Name	Data Type	Data Type Semantics	Description
85	octetTotalCount	unsigned64	totalCounter	The total number of octets in incoming packets for this Flow at the Observation Point since the Metering Process (re-)initialization for this Observation Point. The number of octets includes IP header(s) and IP payload.
166	notSentFlowTotalCount	unsigned64	totalCounter	The total number of Flow Records that were generated by the Metering Process and dropped by the Metering Process or by the Exporting Process instead of being sent to the Collecting Process.
168	notSentOctetTotalCount	unsigned64	totalCounter	The total number of octets in packets in Flow Records that were generated by the Metering Process and dropped by the Metering Process or by the Exporting Process instead of being sent to the Collecting Process. There are several potential reasons for this including resource shortage and special Flow export policies.
150	flowStartSeconds	dateTimeSeconds	default	The absolute timestamp of the first packet of this Flow.
151	flowEndSeconds	dateTimeSeconds	default	The absolute timestamp of the first packet of this Flow.



```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   IEID = 8                  |        Field Length = 4       | 13
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   IEID = 12                 |        Field Length = 4       | 14
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   IEID = 27                 |        Field Length = 16      | 15
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   IEID = 28                 |        Field Length = 16      | 16
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   IEID = 296                |        Field Length = 4       | 17
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   IEID = 150                |        Field Length = 4       | 18
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   IEID = 151                |        Field Length = 4       | 19
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   IEID = 163                |        Field Length = 8       | 20
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   IEID = 85                 |        Field Length = 8       | 21
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   IEID = 166                |        Field Length = 8       | 22
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   IEID = 168                |        Field Length = 8       | 23
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

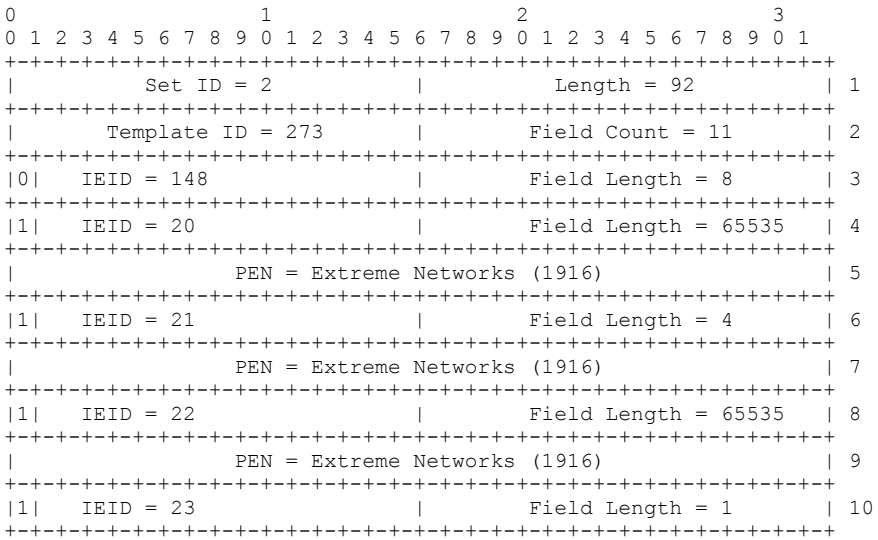
TABLE 11 Template for DNS application

Element ID	Name	Data Type	Data Type Semantics	Description
148	flowId	Unsigned64	Identifier	An identifier of a Flow that is unique within an Observation Domain. This Information Element can be used to distinguish between different Flows if Flow Keys such as IP addresses and port numbers are not reported or are reported in separate records.
60	dnsFlags	unsigned16	default	DNS query/query response flags.
61	dnsQueryType	unsigned16	default	QTYPE picked from DNS Query Response.
62	dnsResponseTime	unsigned64	default	Difference between DNS Response time and DNS Query Request time in nS.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Set ID = 2           |          Length = 36          | 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Template ID = 272       |        Field Count = 4        | 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   IEID = 148                |        Field Length = 8       | 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|   IEID = 60                 |        Field Length = 2       | 4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 PEN = Extreme Networks (1916)                 | 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|   IEID = 61                 |        Field Length = 2       | 6
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 PEN = Extreme Networks (1916)                 | 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|   IEID = 62                 |        Field Length = 8       | 8
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 PEN = Extreme Networks (1916)                 | 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TABLE 12 Template for SSL/TLS Certificates

Element ID	Name	Data Type	Data Type Semantics	Description
148	flowId	Unsigned64	Identifier	An identifier of a Flow that is unique within an Observation Domain. This Information Element can be used to distinguish between different Flows if Flow Keys such as IP addresses and port numbers are not reported or are reported in separate records.
20	sslCertificateSubjectCName	String	default	SSL Certificate Common Name
21	sslCertificateLength	Unsigned32	Identifier	Total length of the SSL Certificate
22	sslFingerPrint	String	Identifier	Short sequence of bytes used to identify public key of the certificate
23	sslVersion	Unsigned8	Identifier	TLS/SSL protocol version used for sharing the certificate
24	sslSerialNumber	String	Identifier	SSL Certificate serial number
25	sslSignatureAlgorithm	String	default	Algorithm Identifier used to sign the SSL Certificate
26	sslIssuerValue	String	default	Value of entity identifier which has signed and issued the SSL Certificate
27	sslValidityNotBefore	String	default	Date on which the certificate validity period begins
28	sslValidityNotAfter	String	default	Date on which the certificate validity period ends
29	sslSubjectValue	String	default	Information associated with the subject of the certificate



```
|                 PEN = Extreme Networks (1916)                 | 11
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|   IEID = 24                 |        Field Length = 65535   | 12
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 PEN = Extreme Networks (1916)                 | 13
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|   IEID = 25                 |        Field Length = 65535   | 14
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 PEN = Extreme Networks (1916)                 | 15
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|   IEID = 26                 |        Field Length = 65535   | 16
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 PEN = Extreme Networks (1916)                 | 17
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|   IEID = 27                 |        Field Length = 65535   | 18
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 PEN = Extreme Networks (1916)                 | 19
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|   IEID = 28                 |        Field Length = 65535   | 20
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 PEN = Extreme Networks (1916)                 | 21
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|   IEID = 29                 |        Field Length = 65535   | 22
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 PEN = Extreme Networks (1916)                 | 23
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TABLE 13 Template for HTTP

Element ID	Name	Data Type	Data Type Semantics	Description
148	flowId	Unsigned64	Identifier	An identifier of a Flow that is unique within an Observation Domain. This Information Element can be used to distinguish between different Flows if Flow Keys such as IP addresses and port numbers are not reported or are reported in separate records.
40	httpHost	String	Identifier	The HTTP request host of the request
41	httpURI	String	Identifier	The HTTP request target of the request
42	httpUserAgent	String	Identifier	The HTTP User-Agent header field
43	httpReferer	String	Identifier	identifies the address of the URI that linked to the current resource being requested

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Set ID = 2           |          Length = 44          | 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Template ID = 271       |        Field Count = 5        | 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|   IEID = 148                |        Field Length = 8       | 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|   IEID = 40                 |        Field Length = 65535   | 4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 PEN = Extreme Networks (1916)                 | 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|   IEID = 41                 |        Field Length = 65535   | 6
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                PEN = Extreme Networks (1916)                | 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|   IEID = 42                |        Field Length = 65535   | 8
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                PEN = Extreme Networks (1916)                | 9
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|   IEID = 43                |        Field Length = 65535   | 10
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                PEN = Extreme Networks (1916)                | 11
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Configuring IPFIX collector

Export of flows can be configured by switching to Flow Exporter Configuration mode and setting the relevant CLI parameters. Clearing this parameter disables IPFIX packets from being sent out. This parameter is disabled by default.

> **NOTE**
> The PEN value for the flow exporter is set to 1916 (for Extreme Networks) by default. This cannot be changed to any other value.

The following properties can be configured for a collector:

TABLE 14 IPFIX parameters

Parameter	Description
name	A string of up to 32 characters to name the collector.
collector-ip	Specifies the IP address of the IPFIX collector.
collector-port	Specifies the remote logical port to be used for UDP transport. Range is 1-65535.
export-ip	Specifies the local interface to be used for export. This IP address must be configured on a vNIC.
export-port	Specifies the local logical port to be used for UDP transport.

A collector can be removed by using `del name` command followed by its name.

Configuring the flow exporter does not initiate generation of IPFIX packets. For IPFIX packets to be generated, the feature must be enabled on the interface and/or in a SMARTMatch rule, by setting the `export-ipfix` CLI parameter.

IPFIX export can be enabled/disabled on the Interface or a SMARTMatch rule irrespective of whether a collector is configured in vTAP. If a collector is not configured, IPFIX export does not occur.

Example - Adding IPFIX collector

```
vtap> en
Password:
vtap# conf t

vtap(config)# flow-exporter

vtap(config-flow-exporter)# add name=collector-1 collector-ip=10.37.43.151 collector-port=4739
transport=udp export-ip=10.37.131.230 export-port=7777
configure collector :successfully.

vtap(config-flow-exporter)# add name=collector-2 collector-ip=10.37.43.152 collector-port=4739
transport=udp export-ip=10.37.131.230 export-port=7788
configure collector :successfully.

vtap(config-flow-exporter)# show flow-exporter config collector=all
Collector ID         : 1
name                 : collector-1
Collector Address    : 10.37.43.151
```

```
Export Address       : 10.37.131.230
Transport Protocol   : UDP
Collector Port       : 4739
Export Port          : 7777
State                : ACTIVE

Collector ID         : 2
name                 : collector-2
Collector Address    : 10.37.43.152
Export Address       : 10.37.131.230
Transport Protocol   : UDP
Collector Port       : 4739
Export Port          : 7788
State                : ACTIVE
```

Example - Deleting IPFIX collector

```
vtap(config-flow-exporter)# del name=collector-1
Flow-export collector deleted successfully.

vtap(config-flow-exporter)# show flow-exporter config collector=all
Collector ID         : 2
name                 : collector-2
Collector Address    : 10.37.43.152
Export Address       : 10.37.131.230
Transport Protocol   : UDP
Collector Port       : 4739
Export Port          : 7788
State                : ACTIVE
```

Example - Displaying templates

```
vtap(config)# show config template=all
Template ID     : 300
        Set ID                          : 3
        Number of Scope Fields          : 2
        Number of Fields                : 8
        Number of Dependent Templates   : 0
Template ID     : 257
        Set ID                          : 2
        Number of Fields                : 24
        Number of Dependent Templates   : 1
                Dependent Template ID   : 300
Template ID     : 256
        Set ID                          : 2
        Number of Fields                : 20
        Number of Dependent Templates   : 1
                Dependent Template ID   : 300
Template ID     : 258
        Set ID                          : 2
        Number of Fields                : 8
        Number of Dependent Templates   : 1
                Dependent Template ID   : 300

vtap(config)# show config collector=all
    Collector ID         : 1
    name                 : test
    Collector Address    : 10.9.9.186
    Export Address       : 10.9.9.184
    Transport Protocol   : UDP
    Collector Port       : 8888
    Export Port          : 9999
```

Enabling IPFIX on the interface

Run the `set export-ipfix` command from Interface Configuration mode to enable IPFIX on the interface.

Example

```
Username: root
Password:

vtap> en
Password:
vtap# conf t

vtap(config)# interface
vtap(config-interface)# edit interface=ip
vtap(config-interface-ip)# show interface=all

interfaces information

                    name  : ip
            traffic type  : ip
                  egress  : drop
    vtap ingress ports  : ens224,ens256
                    mode  : session
            export IPFIX  : disabled

vtap(config-interface-ip)# set export-ipfix=all
Interface "ip"  parameters "export ipfix" updated successfully.
```

Disabling IPFIX on the interface

Run the `clear export-ipfix` command from Interface Configuration mode to disable IPFIX on an interface.

Example

```
vtap(config-interface-ip)# clear export-ipfix
Interface "ip"  parameters "export ipfix" cleared successfully.

vtap(config-interface-ip)# show interface=all

interfaces information

                    name  : ip
            traffic type  : ip
                  egress  : drop
    vtap ingress ports  : ens224,ens256
                    mode  : session
            export IPFIX  : disabled
```

Configuring IPFIX for a SMARTMatch rule

Run the `add rule export-ipfix` command from SMARTMatch Policy Configuration mode to configure IPFIX for a SMARTMatch rule.

Example

```
vtap> en
Password:
vtap# conf t

vtap(config)# smartmatch

vtap(config-smartmatch)# add smatch-policy=smp1
vtap(config-smartmatch-smp1)# add rule protocol=ip export-ipfix=all
Adding rule, action = 'no action'

Rule with Rule ID 1 configured successfully for policy smp1
```

Configuring timeout interval

Another parameter that can be configured for IPFIX is the `export-interval` parameter. This specifies the interval after which the IPFIX metadata transmission is initiated. The default value is one minute.

> NOTE
> It is not recommended to send IPFIX metadata for each Tunnel/Flow at the same time. The export should be for a batch of flows.

Example - Setting export interval to new value

```
vtap(config-flow-exporter)# show export-interval
Time for IPFIX Report Interval : 1 min

vtap(config-flow-exporter)# set export-interval=10
Flow-export report interval updated successfully.

vtap(config-flow-exporter)# show export-interval
Time for IPFIX Report Interval : 10 min
```

Example - Setting export interval to default value

```
vtap(config-flow-exporter)# clear export-interval
Flow-export config updated successfully.

vtap(config-flow-exporter)# show export-interval
Time for IPFIX Report Interval : 1 min
```

SNMP

This section provides information about the SNMP commands supported by vTAP. The commands can be executed using Net-SNMP as a manager.

SNMP is supported for up/down traps running on vTAP and for all other default Linux SNMP statistics.

When you install vTAP, a Management Information Base (MIB) file is installed in the `/usr/share/snmp/mibs` directory. This MIB defines variables and tables that are associated with vTAP.

> **NOTE**
> Use a MIB browser to access the MIB variables. All MIB browsers perform queries and load MIBs.

Few of the SNMP functionality supported in this version of vTAP are:

- Support for SNMPv3 and SNMPv2.
- Support for Entity MIB.
- Trap when the process has started, restarted or stopped.

SNMPWALK commands

This section lists all the snmpwalk commands that vTAP supports.

Setting MIBS environmental variable

After starting the Linux session, before running snmpwalk commands, run the following command to set the MIBS environmental variable:

```
export MIBS=all
```

> **NOTE**
> This command does not display any output.

snmpwalk on Entity MIB

```
[root@vtap]# snmpwalk -v 2c -c public localhost ENTITY
ENTITY-MIB::entPhysicalDescr.1 = STRING: Extreme Virtual TAP (vTAP) is a full-featured network visibility
solution software based tap built for virtualized service provider and enterprise networks.
ENTITY-MIB::entPhysicalVendorType.1 = OID: EXTREME-VTAP-MIB::extremeVirtualTAP
ENTITY-MIB::entPhysicalContainedIn.1 = INTEGER: 1
ENTITY-MIB::entPhysicalClass.1 = INTEGER: other(1)
ENTITY-MIB::entPhysicalParentRelPos.1 = INTEGER: 1
ENTITY-MIB::entPhysicalName.1 = STRING: Extreme Virtual TAP 2.0.0
ENTITY-MIB::entPhysicalHardwareRev.1 = STRING:
ENTITY-MIB::entPhysicalFirmwareRev.1 = STRING:
ENTITY-MIB::entPhysicalSoftwareRev.1 = STRING:
ENTITY-MIB::entPhysicalSerialNum.1 = STRING:
ENTITY-MIB::entPhysicalMfgName.1 = STRING: Extreme Networks
ENTITY-MIB::entPhysicalModelName.1 = STRING: Extreme Virtual TAP 2.0.0
ENTITY-MIB::entPhysicalAlias.1 = STRING:
ENTITY-MIB::entPhysicalAssetID.1 = STRING:
ENTITY-MIB::entPhysicalIsFRU.1 = INTEGER: true(1)
ENTITY-MIB::entPhysicalUris.1 = ""
```