Transfer of Information (TOI)

View All Support

User-defined TCAM profiles TOI

Written by Shyam Kota (/en/support/toi/author/128-shyamk) | Posted on June 14, 2019
Updated on June 1, 2023 | 6414 Views

TCAM (/en/support/toi/tag/tcam) EOS 4.24.1F (/en/support/toi/tag/eos-4-24-1f)
EOS 4.29.2F (/en/support/toi/tag/eos-4-29-2f) EOS 4.30.0F (/en/support/toi/tag/eos-4-30-0f)

Description

This article describes a set of CLI commands to create TCAM profiles. The profile is composed of a set of TCAM features, with each feature having customized lookup key, actions and packet types to hit.

Platform compatibility

- DCS-7280 series
- DCS-7020 series
- DCS-7500 series

Table of Content Hide

- DCS-7800 series

For information about DCS-7050, DCS-7060, CCS-710, CCS-720, CCS-722, and DCS-7010 series, please see User-defined TCAM Profiles (/en/support/toi/eos-4-26-0f/14755-user-defined-tcam-profiles).

Release Support

This feature was introduced in 4.20.5F and will be supported in all future releases.

CLI Command	Release
hardware tcam	4.20.5F
[no|default] profile <newProfile>	4.20.5F
profile <newProfile> copy <srcProfile>	4.20.5F
show [pending] | active | diff	4.20.5F
Abort	4.20.5F
[no|default] feature <name>	4.20.5F
[no|default] feature <name> [copy <srcProf>]	**4.21.3F**
sequence <num>	4.20.5F
(no|default) sequence	4.20.5F
key size limit <size>	4.20.5F
(no|default) key size limit	4.20.5F
[no|default] key field <field1> [<field2> <field3> ... <fieldN>]	4.20.5F

CLI Command	Release
[no|default] action <action1> [<action2>... <actionN>]	4.20.5F
[no|default] packet <packetType>	4.20.5F
[no|default] port qualifier size <num> bits	**4.23.0F**
system profile <profileName>	4.20.5F
(no|default) system profile	4.20.5F
[no|default] feature <name> bank minimum count <count>	**4.21.0F**
[no|default] feature <name> bank maximum count <count>	**4.23.2F**
tap aggregation	4.20.5F
mode exclusive profile <profileName>	4.20.5F
mode mixed module <line card list> [profile <profile>]	**4.21.0F**
show hardware tcam profile	4.20.5F
show hardware tcam profile [<profileName>] detail	4.20.5F
show hardware tcam profile ((<profileName> ((feature <featureName>) detail)) | detail	**4.22.0F**

-- Example – Adding a new Key Field to an existing feature

-- Example – Adding a New Feature to existing profile

-- Configurable Profile Attributes

-- Features

-- Security ACL Features

-- Policy Based Routing Features

-- Tunnels

-- Quality of Service

-- Tap Aggregation

-- Mirroring

-- Sflow

-- MPLS

-- Forwarding Destination Prediction

-- MPLS EVPN

-- Counters

-- Flow-spec

-- DirectFlow

-- IPsec

-- MACsec

-- Traffic Policy

-- Sampled Flow Tracking IPv4 Hardware Offload

-- Postcard Telemetry

-- VRF Selection

-- Miscellaneous features

CLI Command	Release
hardware access-list mechanism (tcam | algomatch | none)	4.20.5F

Overview

TCAM profiles are used to enable TCAM lookups for specific features in the forwarding plane. Due to hardware resource limitations, not all features that use TCAM lookups can be enabled at the same time. The actual set of TCAM lookups that the forwarding chip does for each packet has to be programmed when the forwarding chip is initialized. TCAM banks are another hardware resource that need to be allocated for various features. However, the TCAM banks are allocated on-demand at runtime per feature. Since TCAM lookups have to be programmed when the forwarding chip is initialized, users need to choose what features to enable based on their particular requirements. These feature requirements are grouped together and expressed in terms of TCAM profiles. Hardware resources are also constrained by the number of key fields that can be matched on a feature, the number of actions that can be taken, etc. All of the features, key fields, actions, etc are captured in the TCAM profile.

Due to the constant addition of new features and functionality, arbitrary profiles that used to fit in an old release may no longer fit in a later release. To ensure that this does not happen with any customer

-- Actions

-- Action-Priority

-- Packet Types

-- Key Fields

-- Ethernet Fields

-- Ethernet Internal Fields

-- Encapsulated Ethernet Fields

-- IPv4 Fields

-- IPv4 Internal Fields

-- Encapsulated IPv4 Fields

-- IPv6 Fields

-- IPv6 Internal Fields

-- Encapsulated IPv6 Fields

-- MPLS Fields

-- Vxlan Fields

-- GRE Fields

-- Layer 4 Fields

-- Layer 4 Internal Fields

-- Inner Layer 4 Fields

-- Aegis-specific Fields

-- Miscellaneous Fields

-- User-Defined Fields (UDF) Fields

-- Other TCAM Profile Attributes

-- Syslogs

-- Errors (TCAM-3-PROGRAMMING_FAILURE):

-- Warnings:

-- Non-template warnings:

profile, it is important to work with customer support to ensure that any used customer TCAM profile continues to work in future releases. The list of TCAM profiles that we are aware of and test is captured in the Known TCAM Profiles TOI (/en/support/toi/tcam-profile).

```
    -- Other messages:
    -- Agent Restarts
-- Limitations
-- Future Compatibility
```

Security ACL Mechanism

While this is not directly related to the tcam profile configuration, this configuration results in substantially different hardware setup and is relevant to the tcam profile configuration as well. To enable a set of features, the required tcam profile configuration can be substantially different between AlgoMatch and TCAM access-list modes, so it's important to be aware of the access-list mode of the switch before configuring tcam profiles.

hardware access-list mechanism (tcam | algomatch | none)

- **tcam** - The *`tcam`* keyword specifies to match security acls in TCAM hardware.
- **algomatch** - The *`algomatch`* keyword specifies to match security acls in Algomatch hardware.
- **none** - The *`none`* keyword specifies to use neither tcam nor algomatch hardware for security ACLs. Security ACLs are not supported with the *`none`* keyword.

The following products match security ACLs in Algomatch hardware by default.: DCS-7280QRA, DCS-7280CR2A, DCS-7280CR2K, DCS-7280SR2A, DCS-7280SR2K, DCS-7020TRA.

The following products are configured to match security ACLs in TCAM hardware by default: 7500E, DCS-7280SE, 7500R2, DCS-7280CR2, DCS-7280SR2, DCS-7280SR2M, DCS-7020TR, 7500R3, DCS-7280PR3, DCS-7280DR3, DCS-7280CR3, DCS-7280PR3K, DCS-7280DR3K, DCS-7280CR3K, DCS-7280CR3MK.

All R3 products are in tcam mode only.

The recommendation is to use tcam mode unless the algomatch mode is known to be required.

Configuration

The TCAM profile CLI's configuration is under ***`hardware tcam`*** configuration mode.

```
(config)# hardware tcam
(config-hw-tcam)#
```

Profile Configuration

System profiles are predefined profiles that we ship with Arista EOS. Users can use a predefined system profile or create a TCAM profile to meet specific requirements.

There are two ways to create a TCAM profile. You can create a profile from scratch or create a profile based on an existing system profile. Previously, not all system profiles were copyable, but all profiles that users create using the TCAM profile CLI have always been copyable. Starting in release 4.24.0.F, all profiles are copyable. The recommended way to create a new TCAM profile is to base it on an existing one. The table below indicates the first release that individual system profiles are copyable. To see the features that are enabled in system profiles use the show ***`hardware tcam profile`*** command. For example to see the features in mirroring acl issue the ***`hardware tcam profile mirroring-acl`*** command. To see the features, key fields, actions, etc in a profile append ***`detail`*** to the command. I.e.: ***`hardware tcam profile mirroring-acl detail`***

System Profile	First Supported	First Copyable	Description
default	4.20.5F	4.20.5F	This is the profile that is enabled by default. No configuration is necessary.

System Profile	First Supported	First Copyable	Description
mirroring-acl	4.20.5F	4.20.5F	This profile enables mirroring features. It disables some VXLAN, ACL, PBR, MPLS, and counter features.
mpls-evpn	4.20.5F	**4.24.0F**	This profile enables the MPLS EVPN feature. It disables some QoS and counter features.
pbr-match-nexthop-group	4.20.5F	4.20.5F	This profile enables PBR features. It disables some ACL, mirroring, and counter features.
qos	4.20.5F	**4.21.0F**	This profile enables QoS MAC functionality. It disables some ACL, PBR, mirroring, and counter functionality.
tap-aggregation-default	4.20.5F	4.20.5F	This profile enables TAPAGG functionality.
tap-aggregation-extended	4.20.5F	4.20.5F	This profile includes TAPAGG features provided by tap-aggregation-default, plus: • IPv6 functionality • Tool port truncation • ACL counters • User-defined qualifier matching Caveats: • Disables ACL functionality • Each TCAM rule spans two memory banks. i.e. halves the number of available rules

System Profile	First Supported	First Copyable	Description
tc-counters	4.20.5F	**4.23.0F**	Use this profile to enable tc-counters feature.
vxlan-routing	4.20.5F	4.20.5F	Use this profile to enable VXLAN routing feature.

Note some of these profiles were supported before 4.20.5F. However, since 4.20.5F is the first release that this CLI is supported we are documenting 4.20.5F as the first supported release.

Profile Configuration – existing profile

When you configure a profile based upon an existing profile, all of the configuration in the existing profile is copied to the new profile. From there, the profile can be modified to suit your requirements. In the example below, we copy from the profile ***default***. Although the ***default*** profile suits a wide variety of requirements, it is generally advisable that you should copy from a copyable profile that is closest to your requirements.

```
(config-hw-tcam)# profile <profilename> copy default
(config-hw-tcam-profile-<profile>)#
```

Profile Configuration – from scratch

The other way to create a new profile is from scratch. When you create a profile from scratch, you need to specify all of the features, packet types, actions, key fields, etc. We do not recommend creating profiles from scratch. Since the initial release, we have added commands that make it easier to create profiles based on existing profiles or a profile template. See example sections below for more details.

```
(config-hw-tcam)# profile <profilename>
(config-hw-tcam-profile-<profile>)#
```

In addition, this command can also be used to update existing TCAM profiles. Note, system profiles that are shipped with EOS cannot be modified. Only profiles that users create are able to be modified. If you want to make changes

to a system profile, make a copy of the profile and then make the changes to the copy.

Profile Configuration – from URL

Starting in 4.30.0 support is added to allow loading the contents of a TCAM profile from a local URL on the device. The following configuration is used to load a profile from a URL:

```
(config-hw-tcam)# profile sampleProfile
(config-hw-tcam-profile-sampleProfile)# source ?
  file:   Local URL to load TCAM profile
  flash:  Local URL to load TCAM profile
(config-hw-tcam-profile-sampleProfile)# source flash:presetProfile.txt
(config-hw-tcam-profile-sampleProfile)# exit
```

Upon exit, the URL specified is loaded. If the load is successful, there is a syslog message to indicate success. If the load fails, an error message is printed on the CLI, a syslog message indicating error, and the profile configuration changes are rolled back.

Note: There can be multiple TCAM profiles present inside a URL file. Hence, the name of the profile on the CLI should match with the name of the profile present inside the URL file.

URL file configuration

The file formatting for a URL TCAM profile is the same as CLI. One must start with *'profile <profileName>'* and then proceed to define the features present for the profile. Note that one can have multiple profiles defined in a single file. When specifying the URL in CLI, the name of the profile being configured must be present in the URL configuration file. See the sample URL file based on the above profile definition:

/mnt/flash/presetProfile.txt

```
profile sampleProfile
   feature ...
   ...
profile foo
   feature ...
   ...
```

Syslog messages

Upon successful load of URL the following syslog message is seen:
TCAM-6-PROFILE_IMPORT_SUCCESS: Successfully imported TCAM profile <profileName> from source <URL-path>.

Upon failure to load URL the following syslog message is seen:
TCAM-3-PROFILE_IMPORT_FAILED: Failed to import profile <profileName> from source <URL-path>. Possible reason: <errorMsg>.

Deleting TCAM ProfilesTo remove a profile, use:

```
(config-hw-tcam)# no profile <profilename>
```

Note, system profiles that are shipped with EOS cannot be deleted. Only profiles that users create are able to be deleted.

Applying a Profile

The profile is saved after exiting the profile config mode. To use the newly defined profile, the following CLI command is available to apply the profile to the system globally.

```
(config-hw-tcam)# system profile <profile>
```

```
switch(config-tcam)#system profile default
!
WARNING!
Changing TCAM profile will cause forwarding agent(s) to exit and restart.
All traffic through the forwarding chip managed by the restarting
forwarding agent will be dropped.

Proceed [y/n]
```

Applying a Tap Aggregation Profile

Tap Aggregation profiles are applied differently than other profiles. Tap aggregation profiles are applied under the ***`tap aggregation`*** configuration rather than the ***`hardware tcam`*** configuration.

Applying a profile globally

Users can apply a Tap Aggregation profile globally under the ***tap aggregation*** configuration using the ***mode exclusive profile*** command. With this command, the tap aggregation profile will be applied to tap aggregation line cards. For more information about Tap Aggregation profiles, please refer to the Tap aggregation – user-defined TCAM profiles (/support/toi/eos-4-20-5f/13980-tap-aggregation-user-defined-tcam-profiles) TOI.

```
7500(config)# tap aggregation
7500(config-tap-agg)# mode exclusive profile <profile>
```

Applying per line card Tap Aggregation profiles

To configure different Tap Aggregation tcam profiles on line cards, the ***mode mixed module*** command can be used. It allows specifying a Tap Aggregation profile on a per line card basis. For more information about configuring per line card Tag Aggregation profiles please refer to the Tap Aggregation support per-linecard TCAM profile configuration (/support/toi/eos-4-21-0f/14050-tap-aggregation-support-per-linecard-tcam-profile-configuration) TOI.

```
7500(config)# tap aggregation
7500(config-tap-agg)# mode mixed module <line card list> [ profile <profile> ]
```

For example:

```
7500(config)# tap aggregation
7500(config-tap-agg)# mode mixed module linecard 3 profile tap-aggregation-default
7500(config-tap-agg)# mode mixed module linecard 4 profile tap-aggregation-default
7500(config-tap-agg)# mode mixed module linecard 5,6 profile tap-aggregation-extended
```

TCAM Profile Rollback Support

The TCAM profile rollback feature will automatically restore the last programmed TCAM profile in the case of programming failure. A programming failure is when the configured profiles fails to program on one or more linecards or fails to program on a fixed system. There is two case where TCAM rollback would be performed:

- If updating a currently programmed TCAM profile fails, the previous version of the running TCAM profile will be restored.
- If switching to a different TCAM profile fails, the previous TCAM profile will be restored.

Troubleshooting

If the TCAM profile configuration is rollbacked due to programming the failure, the programming status and errors will be display in the cli:

```
switch(config-tcam)#system profile bad-profile
!
WARNING!
Changing TCAM profile will cause forwarding agent(s) to exit and restart.
All traffic through the forwarding chip managed by the restarting
forwarding agent will be dropped.

Proceed [y/n]y
                  Configuration         Status
Linecard3         bad-profile           ERROR
Linecard4         bad-profile           ERROR
Linecard5         bad-profile           ERROR
Linecard6         bad-profile           ERROR

Detailed Programming Status:

Linecard3, Linecard4, Linecard5, Linecard6
   [Error] the key size of feature 'acl port ip' exceeds the configured key
size limit
```

Release History

Release	Update
4.27.1F	Initial introduction

Feature Configuration

Features can be added or deleted from the new profile.

```
(config-hw-tcam-profile-<profile>)# [no] feature <feature>
```

The features are described by hierarchical CLI keywords. For example, IPv4 port ACL is represented by ***feature acl port ip*** and IPv6 port ACL is represented by ***feature acl port ipv6***. Under each feature mode, there are various fields that can be modified.

It is not recommended that users create features from scratch. Creating features from scratch is a lot of configuration and it is easy to make mistakes. Instead it is recommended that users copy features from existing profiles or the profile template. See the ***Feature copy*** section below for more details.

Feature copy

Adding a feature from scratch could be time consuming. When creating a feature from scratch, all required packet types, actions, key fields, etc. must be added to the feature. An alternative to creating the feature from scratch is to create the feature based upon an existing profile. Then the feature can simply be used as is or modified to meet some specific requirements. The following is an example of copying the feature ***qos mac*** from the ***system-feature-source-profile*** profile template. This profile template fully specifies all features with packet types, key fields and actions.

```
yo668(s1)(config)# hardware tcam
yo668(s1)(config-hw-tcam)# profile newfeature copy default
yo668(s1)(config-hw-tcam-profile-newfeature)# feature qos mac copy system-feature-source-profile
```

Configuring Feature Packet Types

This describes packet types that the feature will be applied on. The CLI is

```
(config-hw-tcam-profile-<profile>-feature-<feature>)# [no] packet <packet header keywords> forwarding <bridged|routed|mpls> [multicast] [decap]
```

The packet header is described as a series of CLI packet header keywords after **packet** keyword. It starts from the outermost header after Ethernet. For example, a regular IPv4 packet is ***packet ipv4*** and a VXLAN packet is ***packet***

ipv4 vxlan eth ipv4. The forwarding keyword indicates the forwarding type of the packet. **Multicast** indicates if the packet is a multicast packet. Lastly, **decap** indicates if the packet is decapsulated after tunnel termination.

Configuring Feature Key Fields

This describes the TCAM key format for the feature. The CLI can add or delete fields that are used to build the key.

```
(config-hw-tcam-profile-<profile>-feature-<feature>)# [no] key field <field>
```

All supported key fields can be found with '***key field ?***'

Configuring Feature Key Size

This describes the TCAM key size limit. If too many key fields or actions are added to the feature so that the key size goes beyond the limit, a Syslog will be issued and the profile will fail to apply. The default key size limit is 320 bits. A 320-bit key uses double-wide tcam banks and a 160-bit key uses a single tcam bank.

```
(config-hw-tcam-profile-<profile>-feature-<feature>)# [no] key size limit <size>
```

The number of TCAM entries that can be supported by a feature varies with the width of the TCAM key. A 320-bit key consumes twice the resources as a 160-bit key. A 320-bit key, consumes a minimum of two banks and as the number of entries in the TCAM grows beyond the bank depth, additional banks are allocated two at a time.

For some features, using a small 80-bit key allows for enhanced TCAM utilization. For these features, twice the number of entries are available than if a 160-bit key is used. Key sizes are limited to 80, 160, and 320 bit keys sizes. Shrinking keys sizes won't increase TCAM utilization until a smaller key size is used. However, there are additional hardware constraints such that removing key fields may allow a profile that previously didn't fit in hardware to fit.

Supported TCAM key sizes

Size	Banks Used	Relative entries available
80	1/2	Twice the number of a entries as a 160-bit bank (for some features)
160	1	The number of entries in a bank
320	2	Half the number of entries available

Configuring Feature Actions

This lists the actions to include for this feature. The CLI is

```
(config-hw-tcam-profile-<profile>-feature-<feature>)# [no] action <action>
```

The supported actions can be found through `action ?`.

Configuring Feature Sequence

This describes the programming order that the software uses to program the feature. Changing the order may affect the programming status of a profile. The default sequence is 255. Typically, this doesn't have to be changed.

```
(config-hw-tcam-profile-<profile>-feature-<feature>)# [no] sequence <sequence>
```

Configuring Feature Port Qualifier Size

This describes the qualifier width in bytes that the feature's qualifier field will be dynamically sized to be configured with. Size limits will vary according to TCAM usage. The default size and support is per feature basis. More information can be found in the Support for Greater Than 29 Unique ACLs per Chip with Configurable TCAM Qualifier Sizing (/en/support/toi/eos-4-23-0f/14373-support-for-greater-than-29-unique-acls-per-chip-with-configurable-tcam-qualifier-sizing) TOI. The CLI is

```
(config-hw-tcam-profile-<profile>-feature-<feature>)# [no] port qualifier size <number> bits
```

If N bits are assigned for a feature, the tcam entries for that feature reserves N bits for the ID space, so we could configure upto 2^N unique tcam rulesets for that feature. The specific feature may have a limit lower than 2^N.

Reserving TCAM banks

TCAM banks are a limited resource. If TCAM banks are used by one feature, there may not be enough banks available for another feature. In general, TCAM banks are allocated to features on a first come first served basis. This could sometimes result in a feature occupying more banks than desired with misconfigurations. To provide additional protection for some critical features, users can reserve banks for particular features using the "***feature <feature> bank minimum count <count>***" command. By reserving TCAM resources, those resources are dedicated to a particular feature regardless of whether that feature uses them or not. For more information, please refer to TCAM Reservation on 7500R/ 7280R series (/en/support/toi/eos-4-20-5f/14022-tcam-reservation-on-sand) TOI.

To specify a minimum number of banks that will be reserved for feature:

```
yo668(s1)(config)# hardware tcam
yo668(s1)(config-hw-tcam)# feature <feature> bank minimum count <count>
```

If there is a TCAM feature that may consume lots of TCAM banks due to having lots of ACL ACEs for example, users can limit the number of TCAM banks that a feature uses by specifying a maximum bank usage by the "***feature <feature> bank maximum count <count>***" command. With this command, banks are still flexibility allocated, but it prevents one or more features from using all of the banks.

To specify a maximum number of banks that can be used by a feature:

```
yo668(s1)(config)# hardware tcam
yo668(s1)(config-hw-tcam)# feature <feature> bank maximum count <count>
```

The number of TCAM entries available to features differs by the type of forwarding plane.

Card Type / System	Number Large Banks	Large Bank Size	Number Small Banks	Small Bank Size
7500E, DCS-7280SE	12	1024	2	128
7500R, DCS-7280QR, DCS-7280QRA, DCS-7280CR	12	2048	4	128
7500R2, DCS-7280CR2, DCS-7280CR2A, DCS-7280CR2K, DCS-7280SR2, DCS-7280SR2M, DCS-7280SR2A, DCS-7280SR2K	12	2048	4	128
DCS-7020TR, DCS-7020TRA	12	1024	4	128
7500R3, DCS-7280PR3, DCS-7280PR3K, DCS-7280DR3, DCS-7280DR3K, DCS-7280CR3, DCS-7280CR3K, DCS-7280CR3MK	12	2048	4	256

Show Commands

A show command is available to list the TCAM profile status on each line card.

```
(config)# show hardware tcam profile
                 Configuration     Status
FixedSystem      newprofile1       newprofile1
```

If the profile cannot be programmed, the Status column will print ‘ERROR‘. Any features that use TCAM functionality won’t work properly. Do not expect any features to work if the profile is in the ‘ERROR’ state. See Limitations for more information. If there are warnings or errors, a summary message will display warnings or errors found in programming the profile in addition to the system log messages.

```
(config)# show hardware tcam profile
                     Configuration        Status
Linecard3            newprofile1          ERROR
Linecard4            newprofile1          ERROR
Linecard5            newprofile1          ERROR
Linecard6            newprofile1          ERROR
Linecard7            newprofile1          WARNING

Detailed Programming Status

Linecard3, Linecard4, Linecard5
   [Error] feature flow is not supported on this hardware platform

Linecard7
   [Warning] the key size of feature flow exceeds the configured key size li
mit
```

The content of a TCAM profile can be displayed with:

```
(config)# show hardware tcam profile <profile> [detail]
```

For example, the output for ***`newprofile1`*** is:

```
(config-hw-tcam)# show hardware tcam profile newprofile1 detail
Profile newprofile1 [ FixedSystem ]
 Feature         mpls
 --------------- ------------------------------------------------------------
-
 Key size        160
 Actions         drop, redirect, set-ecn
 Packet type     ipv4 mpls ipv4 forwarding mpls decap
                 ipv4 mpls ipv6 forwarding mpls decap
                 mpls ipv4 forwarding mpls
                 mpls ipv6 forwarding mpls
                 mpls non-ip forwarding mpls

 Feature         acl vlan ipv6
 --------------- ------------------------------------------------------------
-
 Key size        320
 Key fields      dst-ipv6, ipv6-next-header, l4-dst-port, l4-src-port,
                 src-ipv6-high, src-ipv6-low, tcp-control
 Actions         count, drop, mirror, redirect
 Packet type     ipv6 forwarding routed
...
```

Note that the profile contains all the features that are untouched after copying from the base profile.

If a user wants to see all of the features configured in a profile without seeing how the features are defined, the user can issue the ***`show hardware tcam profile`*** command without the ***`detail`*** keyword.

```
(config-hw-tcam-profile-newfeature)# show hardware tcam profile default
Features enabled in TCAM profile default: [ Linecard3, Linecard4, Linecard6,
Linecard7, Linecard8, Linecard9, Linecard10 ]
 mpls
 acl vlan ipv6
 acl subintf ipv6
 acl vlan ipv6 egress
 acl port ipv6
 pbr ipv6
 acl vlan ip
 acl subintf ip
 acl port ip
 tunnel vxlan
 acl port mac
 pbr ip
 pbr mpls
 qos ipv6
 qos ip
 mirror ip
 counter lfib
 mpls pop ingress
```

Examples

Example – Adding a new Key Field to an existing feature

This example demonstrates how to create a new profile to match on a vlan field in MAC ACL. The ***`MAC ACL`*** feature exists in the ***`default`*** profile. Since the new ***`macvlan`*** profile is copied from the ***`default`*** profile, it inherits all of the features from the ***`default`*** profile. The feature ***`acl port mac`*** is contained in the ***`default`*** profile. So all of the key fields, packet types, etc. configured in the ***`default`*** profile are configured in the ***`macvlan`*** profile identically to the ***`default`*** profile. Consequently, the only thing that has to be configured in this feature is adding the ***`vlan`*** key field. If the ***`acl port mac`*** feature wasn't present in the profile from which it was copied, the feature would either have to be created from scratch or copied from a different profile. Sometimes, adding a key field will require removal of a different key field to fit within the tcam width restrictions.

```
(config-hw-tcam)# profile macvlan copy default
(config-hw-tcam-profile-macvlan)# feature acl port mac
(config-hw-tcam-profile-macvlan-feature-acl-port-mac)# key field vlan
(config-hw-tcam-profile-macvlan-feature-acl-port-mac)# exit
(config-hw-tcam-profile-macvlan)# exit
Saving new profile 'macvlan'
(config-hw-tcam)# system profile macvlan
```

Example – Adding a New Feature to existing profile

The recommended way to add a feature to a new TCAM profile is to copy the feature from an existing profile. EOS contains a special template profile that contains representative configuration for all configurable features. You use this template profile to add features to your new profile. If the system template profile does not define the features according to a particular customer requirement, those features should be modified after they are copied. Additionally, features can be copied from other profiles as well.

```
yo668(s1)(config)# hardware tcam
yo668(s1)(config-hw-tcam)# profile newfeature copy default
yo668(s1)(config-hw-tcam-profile-newfeature)# feature qos mac copy system-feature-source-profile
yo668(s1)(config-hw-tcam-profile-newfeature-feature-qos-mac)# exit
yo668(s1)(config-hw-tcam-profile-newfeature)# exit
Saving new profile 'newfeature'
yo668(s1)(config-hw-tcam)# system profile newfeature
yo668(s1)(config-hw-tcam)#
```

Configurable Profile Attributes

Users create TCAM profiles because it is impossible for the hardware to support all features, key fields, actions, etc concurrently. TCAM profiles are used to select what functionality is actually needed from TCAM resources to support functionality on the switch. Many times the initial list of features and functionality is greater than what can be expressed in the hardware concurrently. Sometimes functionality that is included in a profile is not being used by the configuration and can be removed without impacting the configuration. Removing this unused TCAM functionality may allow a previously unfittable profile to fit. Sometimes, however, it is a trade-off and some desired functionality has to be removed to make room for higher priority functionality requirements.

Features

TCAM features are used to describe some high level router functionality such as security ACLs matching on IPv4 packets or policy based routing on IPv6 packets. When creating a TCAM profile, start with an existing TCAM profile

with a feature set that most closely matches your desired feature set. From there, add any missing TCAM features. If the profile fits, then you are done. Otherwise, remove any features that the profile isn't using to try to make it fit.

Security ACL Features

Security ACL features' primary purpose is to drop packets that match particular criteria. Security ACLs are enabled with different TCAM features depending on what type of packet is being matched, what type of interface the security ACL is applied on, etc.

- **acl vlan ip** (Ingress IPv4 RACLs applied on SVIs)
  - This feature enables ingress security acls on routed IPv4 packets. Enable this feature when IPv4 security acls are applied on SVI interfaces.

- **acl vlan ipv6** (Ingress IPv6 RACLs applied on SVIs)
  - This feature enables ingress security acls on routed IPv6 packets. Enable this feature when IPv6 security acls are applied on SVI interfaces.

- **acl vlan ipv6 egress (Egress v6 RACLs applied on SVIs)**
  - This feature enables egress security acls on routed IPv6 packets. Enable this feature when IPv6 security acls are applied on SVI interfaces.

- **acl subintf ip** (Ingress IPv4 RACLs applied on subinterfaces)
  - This feature enables ingress security acls on routed IPv4 packets. Enable this feature when IPv4 security acls are applied on subinterfaces.
  - This feature is not required for subinterface acls to work in R3 systems. Use the "acl port ip" feature instead.

- **acl subintf ipv6** (Ingress IPv6 RACLs applied on subinterfaces)
  - This feature enables ingress security acls on routed IPv6 packets. Enable this feature when IPv6 security acls are applied on subinterfaces.
  - This feature is not required for subinterface acls to work in R3 systems. Use the "acl port ipv6" feature instead.

- **acl subintf mac** (Ingress MACLs applied on subinterfaces)
  - This feature enables ingress security acls on bridged Ethernet frames. Enable this feature when Ethernet security acls are applied on subinterfaces.
  - This feature is not required for subinterface acls to work in R3 systems. Use the “acl port mac” feature instead.
- **acl port ip** (Ingress IPv4 PACLs applied on physical interfaces)
  - This feature enables ingress security acls on routed and bridged IPv4 packets. Enable this feature when IPv4 security acls are applied on a port.
- **acl port ipv6** (Ingress IPv6 PACLs applied on physical interfaces)
  - This feature enables ingress security acls on routed and bridged IPv6 packets. Enable this feature when IPv6 security acls are applied on port.
- **acl port mac** (Ingress MACLs applied on physical interfaces)
  - This feature enables ingress security acls on bridged and routed Ethernet frames. Enable this feature when Ethernet security acls are applied on a port.
- **acl port ip egress** (Egress IPv4 PACLs applied of physical interfaces)
  - This feature enables egress security acls on routed and bridged IPv4 packets. Enable this feature when egress IPv4 security acls are applied on a port. For historical reasons, this feature is implicitly added to TCAM profiles by default. Since this feature is implicitly added, issuing “no acl port ip egress” does not delete this feature. To remove this feature, add “acl port ip egress disabled” to the profile.
  - This feature can be explicitly configured by providing key fields, actions and packet types just like other features on R3 Systems. It is first supported in 4.29.2F
- **acl port ip egress disabled** (Disable the “***`acl port ip egress`***”)
  - The “***`acl port ip egress`***” is implicitly added to the profile, to disable this feature add “acl port ip egress disabled” to the profile. See, “acl port ip egress” above for more details.

- **acl port ip egress mpls-tunnelled-match**
  - This feature enables matching on the IPv4 TCP established, IPv4 IP fragmentation, and the L4 Ops 2b fields in the IPv4 header for IPv4 packets when the IPv4 packet is encapsulated over MPLS. This feature does not use additional TCAM banks.

- **acl port ipv6 egress (Egress IPv6 PACLs applied on physical interfaces)**
  - This feature enables egress security acls on routed and bridged IPv6 packets. This feature does use additional TCAM banks.

- **acl gre tunnel interface ip (GRE ACL matching on IPv4 packets)**
  - This feature enables matching on inner IPv4 headers on GRE tunnel interfaces.

- **acl tunnel interface ip**
  - This feature enables matching on inner IPv4 headers on GRE tunnel interfaces, IPIP tunnel interfaces and IPSec tunnel interfaces.

- **acl port mac egress**
  - This feature enables egress MAC security ACLs on bridged and routed Ethernet frames
  - For details on how to use egress MAC security ACLs please refer to the following TOI (/en/support/toi/eos-4-25-0f/14611-egress-mac-acls)

Security ACLs

Feature	Packet Type	Forwarding Type	Interface Type	Direction
acl vlan ip	IPv4	Routed	SVI	Ingress
acl vlan ipv6	IPv6	Routed	SVI	Ingress
acl subintf ip	IPv4	Routed	Subinterface	Ingress
acl subintf ipv6	IPv6	Routed	Subinterface	Ingress
acl subintf mac	Ethernet	Bridged	Subinterface	Ingress
acl port ip	IPv4	Bridged and Routed	Physical port	Ingress

Feature	Packet Type	Forwarding Type	Interface Type	Direction
acl port ipv6	IPv6	Bridged and Routed	Physical port	Ingress
acl port mac	Ethernet	Bridged and Routed	Physical port	Ingress
acl vlan ipv6 egress	IPv6	Bridged and Routed	SVI	Egress
acl port ip egress	IPv4	Bridged and Routed	Physical port	Egress
acl port ipv6 egress	IPv6	Bridged and Routed	Physical port	Egress
acl port ip egress mpls-tunnelled-match	MPLS IPv4		Tunnel	Egress
acl gre tunnel interface ip	GRE IPv4	Routed	Tunnel	Ingress
acl tunnel interface ip	GRE/IPIP/IPSec IPv4	Routed	Tunnel	Ingress

Policy Based Routing Features

Policy-based routing (PBR) is a feature that is applied on routable ports, to preferentially route packets. Forwarding is based on a policy that is enforced at the ingress of the applied interface and overrides normal routing decisions. For additional information see the Policy-based Routing (/en/support/toi/eos-4-23-2f/14429-policy-based-routing) TOI.

- **pbr ip** (Policy based routing matching on IPv4 packets)
  - This ***`pbr ip`*** feature overrides the routing decision on IPv4 packets based on a physical port.

- **pbr ipv6** (Policy based routing matching on IPv6 packets)
  - This ***`pbr ipv6`*** feature overrides the routing decision on IPv6 packets based on a physical port.

- **pbr mpls** (Policy based routing matching on MPLS packets)
  - This ***`pbr mpls`*** feature overrides the routing decision on MPLS packets based on a physical port.

- **pbr subintf ip** (Subinterface PBR policy matching on IP packets)
  - This ***`pbr subintf ip`*** feature overrides the routing decision on IPv4 packets based on a Subinterface. This feature is not required for PBR on subinterfaces to work in the R3 systems. This feature enables up-to 1k unique PBR policies on sub-interfaces.

- **pbr subintf ipv6** (Subinterface PBR policy matching on IPv6 packets)
  - This ***`pbr subintf ipv6`*** feature overrides the routing decision on IPv6 packets based on a Subinterface. This feature is not required for PBR on subinterfaces to work in the R3 systems. This feature enables up-to 1k unique PBR policies on sub-interfaces.

- **pbr subintf mpls** (Subinterface PBR policy matching on MPLS packets)
  - This ***`pbr subintf mpls`*** feature overrides the routing decision on MPLS packets based on a Subinterface. This feature is not required for PBR on subinterfaces to work in the R3 systems. This feature enables up-to 1k unique PBR policies on sub-interfaces.

- **pbr tunnel interface ip** (Tunnel interface PBR policy matching on IP packets)
  - This ***`pbr tunnel interface ip`*** feature overrides the routing decision on inner IP header of the GRE packet getting tunnel decapsulated on a GRE tunnel interface on R3 systems. This feature enables up-to 1k unique PBR policies on GRE Tunnel Interfaces.

Tunnels

Tunnels are primarily used to transparently connect two disjoint networks over a routed IP network. Tunnels encapsulate packets in an IP packet and delivered to a destination.

- **tunnel ipv4** (Tunnel encapsulation on ipv4 traffic)
  - The ***`tunnel ipv4`*** feature allows IP packets to be encapsulated over an IP network. This is needed for any IP-in-IP tunnels like GRE.

- **tunnel vxlan** (Tunnel encapsulation for Virtual Extensible Local Area Network)
  - The ***`tunnel vxlan`*** feature allows Layer 2 Ethernet frames to be encapsulated over an IP/UDP network.
- **tunnel vxlan multicast mlag**
- **tunnel vxlan multi-homing**
  - This feature is required for VXLAN packet forwarding on multi-homed interfaces
- **tunnel vxlan multi-homing domain remote**
- **tunnel vxlan routing** (Tunnel encapsulation for routing VXLAN)
  - The ***`tunnel vxlan routing`*** feature provides IP routing between VXLAN VNIs. Starting from the R3 series of switches, this feature is not required for vxlan routing to work.
- **tunnel vxlan routing v6-underlay**
  - The ***`tunnel vxlan routing v6-underlay`*** feature allows bridging of L2 Ethernet frames and IP routing between VXLAN VNIs over IPv6/UDP network

Quality of Service

Quality of Service defines a method of differentiating data streams to provide varying levels of service to the different streams. Criteria determining a packet's priority level include packet field contents and the port where data packets are received. QoS settings are translated into traffic classes, which are then used to manage all traffic flows. Primarily the actions performed by QoS are setting the DSCP, TC, and policer. For more information about QoS please refer to the product documentation chapter on QoS (/en/um-eos/eos-quality-of-service#xx1097818).

- **qos ip** (Quality of Service on IPv4 traffic)
  - The ***`qos ip`*** feature enables applying Quality of Service on IPv4 packets
- **qos ipv6** (Quality of Service on IPv6 traffic)
  - The ***`qos ipv6`*** feature enables applying Quality of Service on IPv6 packets.

- **qos subintf ip** (Quality of Service on IPv4 traffic on a Subinterface)
  - The ***qos subintf ipv4*** feature enables applying Quality of Service on IPv4 packets on subinterfaces. This feature is not required for subinterface qos to function starting from R3 series of switches. This feature enables up-to 1k unique QOS policies on sub-interfaces.

- **qos subintf ipv6** (Quality of Service on IPv6 traffic on a Subinterface)
  - The ***qos subintf ipv6*** feature enables applying Quality of Service on IPv6 packets on subinterfaces. This feature enables up-to 1k unique QOS policies on sub-interfaces. This feature is not required for subinterface qos to function starting from R3 series of switches.

- **qos mac** (Quality of Service on IPv4 traffic on an Ethernet interface)
  - The ***qos mac*** feature enables applying Quality of Service on Ethernet frames. This feature is not required for subinterface qos to function starting from R3 series of switches.

Quality of Service

Feature	Packet Type	Forwarding Type	Interface Type
qos ip	IPv4	Routed & MPLS	
qos ipv6	IPv6	Routed	
qos subintf ip	IPv4	Bridged and Routed	Subinterface
qos subintf ipv6	IPv6	Bridged and Routed	Subinterface
qos mac	Ethernet	Bridged, MPLS, and Routed	

Tap Aggregation

Ethernet based switches are commonly deployed in dedicated networks to support tap and mirror port traffic towards one or more analysis applications. Ports configured to mirror data can simultaneously switch traffic to its primary destination while directing a copy of that traffic to analysis or test devices. Tap ports are typically part of a dedicated environment that allows for the aggregation of data streams from multiple sources that can be directed to multiple destinations. For more information about configuring Tap Aggregation please refer to the Tap Aggregation (/en/um-eos/eos-test-access-point-aggregation) section of the EOS manual.

- **tapagg ip**
  - This feature enables Tap Aggregation traffic steering on IPv4 traffic.
- **tapagg ipv6**
  - This feature enables Tap Aggregation traffic steering on IPv6 traffic.
- **tapagg mac**
  - This feature enables Tap Aggregation traffic steering on the Ethernet MAC header.
- **tapagg tunnel termination**
  - This feature allows you to decapsulate GRE traffic received on an L3 router port and steer it to a set of tool ports.
  - For details on how to use Tap Aggregation tunnel termination please refer to the following TOI (/support/toi/eos-4-25-2f/14693-support-for-tapagg-gre-tunnel-termination).

Mirroring

Port mirroring, also known as port monitoring, is the duplication of traffic from a collection of source ports to a destination port.

Mirroring allows for configuring IPv4, IPv6, and MAC access lists and then mirrors the packets that match permit statements and does not select packets that match deny statements. The GRE tunnel interfaces act as a logical interface that performs GRE encapsulation or decapsulation. Mirroring is supported on GRE as well. For more information please see the Advanced Mirroring Features (/en/support/toi/eos-4-25-0f/14602-advanced-mirroring-features) TOI.

- **mirror ip**
  - This feature enables mirroring on IPv4 traffic.
- **mirror ipv6**
  - This feature enables mirroring on IPv6 traffic.
- **mirror mac**
  - This feature enables mirroring on MAC traffic.

- **mirror mpls-over-gre ip**
  - This feature enables mirroring on Generic Routing Encapsulation (GRE) tunneling which supports the forwarding over IPv4 GRE tunnel interfaces.

- **mirror mpls-over-gre ipv6**
  - This feature enables mirroring on Generic Routing Encapsulation (GRE) tunneling which supports the forwarding over IPv6 GRE tunnel interfaces.

- **mirror port egress rate-limit**
  - Rate limiting of mirrored traffic provides support to control the rate of mirrored traffic that can egress the switch.
  - On the R and R2 series of switches this feature is required to apply the mirror rate limit configuration to TX mirror sources (without this feature, only RX mirror sources are subject to the rate limiting).
  - For details on how to use mirror rate limiting please refer to the following TOI (/support/toi/eos-4-25-2f/14678-advanced-mirroring-features/#Rate_Limiting_of_Mirrored_Traffic).

Sflow

sFlow is a sampling technique which monitors the incoming traffic on all the interfaces without affecting the network performance. It samples the traffic based on the sampling rate defined globally through a CLI ("***sflow sample <rate>***" or "***sflow hardware acceleration sample <rate>***").

- **sflow subintf**
  - This feature enables ingress sFlow on subinterfaces.

MPLS

Additional information about configuring MPLS can be found in the Multiprotocol Label Switching (MPLS) (/en/um-eos/eos-multiprotocol-label-switching-mpls) section of the EOS documentation.

- **mpls**
  - This feature enables MPLS forwarding and routing.

- **mpls pop ingress**

    - This feature enables ability to apply egress IP ACLs on the inner MPLS payload after the pop label operation.

- **hash mpls-over-gre inner ipv4**
    - This feature enables ecmp hashing on the inner IPv4 header for MPLS over GRE transit traffic.
    - Works with ‘packet ipv4 mpls ipv4 forwarding routed’

- **hash mpls-over-gre inner ipv6**
    - This feature enables ecmp hashing on the inner IPv6 header for MPLS over GRE transit traffic.
    - Works with ‘packet ipv4 mpls ipv6 forwarding routed’

Forwarding Destination Prediction

Forwarding destination prediction enables visibility into how a packet is forwarded through the switch, allowing you to determine which interfaces a packet would egress from. Please refer to the feature TOI (/en/support/toi/eos-4-22-1f/14262-forwarding-destination-prediction) for more information.

- **forwarding-destination**
    - On the R and R2 series of switches this feature is required to use forwarding destination prediction.

- **forwarding-destination mpls**
    - On the R and R2 series of switches, this feature enables the forwarding destination CLI to predict destinations for MPLS routed packets for all specified packet types.

MPLS EVPN

The following TCAM features can be used when using Ethernet VPN with MPLS underlay deployments.

- **evpn mpls irb**
    - This feature is required for EVPN MPLS IRB (/en/support/toi/eos-4-21-3f/14137-evpn-irb-with-mpls-underlay) to work.

- **evpn mpls multi-homing**

- This feature is required for EVPN MPLS multihoming (/support/toi/eos-4-20-5f/13969-l2evpn-mpls) to work.

Counters

The following hardware counter features need TCAM profile support on DCS-7020, DCS-7280R, DCS-7280R2, DCS-7500R and DCS-7500R2 series.

- **counter multicast ipv4**
  - This feature enables the counting of IPv4 multicast routed packets and total octets per multicast route. It is not necessary to configure this feature starting with the R3 series.

- **counter lfib**
  - This feature enables the counting of MPLS LFIB packets and bytes. It is not necessary to configure this feature starting with the R3 series.

- **counter vlan-interface ingress**
  - This feature enables the counting of packets/octets ingressing through the VLAN interface for every VLAN interface configured in the system.

- **counter vlan traffic-class**
  - This feature enables the counting of packets/octets ingressing through the VLAN in per-VLAN per-traffic class mode for every VLAN configured in the system.

- **counter vtep decap**
  - This feature enables the counting of packets/octets coming from the core, decapsulated on the device and heading towards the edge per each VTEP. It is not necessary to configure this feature starting with the R3 series.

Flow-spec

The feature enables carrying of Flow specification "Flowspec" rules and corresponding actions over BGP. These rules are applied on ingress routed traffic. For more information about the BGP Flowspec feature please refer to the BGP Flowspec TOI (/en/support/toi/eos-4-23-2f/14437-support-for-bgp-flowspec-release-updates).

- **flow-spec port ipv4**
  - This feature enables flow-spec for IPV4 traffic on a physical interface.

- **flow-spec port ipv6**
  - This feature enables flow-spec for IPV6 traffic on a physical interface.

DirectFlow

DirectFlow exposes the underlying forwarding ASICs capabilities through a programmable interface like EAPI or the standard CLI.

DirectFlow works in conjunction with all other aspects of standard L2/L3 bridging or forwarding, and DirectFlow traffic is subject to the standard packet processing pipeline within the ASIC. DirectFlow acts like a stage in packet processing that processes traffic after ingress checks and before any egress actions. This feature enables you to configure flows that consist of a matching criteria and actions, and to modify how traffic is processed.

- **flow**
  - This feature enables DirectFlow on IPV4 traffic.

IPsec

Internet Protocol Security (IPsec) is a protocol suite for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a communication session. IPsec includes protocols for establishing mutual authentication between agents periodically during the session and negotiation of cryptographic keys to be used during the session. IPsec supports network-level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection. For more information about configuring IPsec please refer to the Internet Protocol Security (IPsec) (/en/um-eos/eos-data-plane-security#xx1007240) section of the EOS documentation.

- **ipsec** - This feature is only supported on the DCS-7020SRG platform.
  - The ipsec feature enables IPsec on the DCS-7020SRG platform. The IPSec Tunnels on EOS (/support/toi/eos-4-22-0f/14302-ipsec-tunnels-on-eos) TOI provides additional information on configuring IpSec on the DCS-7020SRG platform.

- **ipsec egress padding-removal** - This feature is only supported on the DCS-7020SRG platform.
  - This feature enables the removal of extra padding in small Ethernet frames before being processed by the IPSec hardware engine. This may

be required when interoperating with some non-DCS-7020SRG IPSec peers.

MACsec

Media Access Control Security (MACsec) is an industry standard encryption mechanism that protects all traffic flowing on the Ethernet links. MACsec is based on IEEE 802.1X and IEEE 802.1AE standards. For more information about configuring MACsec please refer to Media Access Control Security (/en/um-eos/eos-data-plane-security#xx1210966) in the EOS documentation.

- **macsec-proxy**
  - This feature enables MACsec service for non MACsec capable front panel ports. MACsec over the front panel ports is provided by mapping a front panel port to a MACSec proxy subinterface. Packets transmitted on the MACsec proxy subinterface will be encrypted and patched to the front panel port. Packets received on the front panel port will be patched to the MACsec proxy subinterface where these are decrypted and then routed. MKA negotiates and renews encryption keys. A MACsec capable front panel port has to be dedicated for this purpose and cannot be plugged in as it will be used to recycle packets being encrypted and decrypted. For more information about this feature please see the MACsec Proxy For Front Panel Ports (/en/support/toi/eos-4-23-2f/14444-macsec-proxy-for-front-panel-ports) TOI.

Traffic Policy

The traffic policy feature can be broken down into two major categories, packets that are destined for the CPU and other forwarded packets. In general traffic policy is used to filter or police traffic. It can be used to protect the device from unwanted traffic flows.

The ***`traffic-policy cpu`*** feature adds support for CPU traffic policy capable of filtering IP traffic which would otherwise hit the CPU. This policy is capable of permitting traffic from trusted sources, while rejecting untrusted traffic. To prevent a malicious source from overloading the CPU, this traffic policy will take effect in the switching hardware, before the traffic is processed by the kernel. This feature provides a shorthand syntax for securing L3 protocol peers, in particular, BGP neighbors. The CPU Traffic Policy TOI (/en/support/toi/eos-4-

22-0f/14201-support-for-cpu-traffic-policy) provides additional information about configuring CPU Traffic Policy. The Traffic Policy on Interfaces TOI (/en/support/toi/eos-4-24-2f/14550-support-for-traffic-policy-on-interfaces) provides additional information about configuring Traffic Policy for other forwarded packets.

- **traffic-policy cpu ipv4**
  - This feature adds support for traffic policy on IPv4 traffic destined to the CPU.

- **traffic-policy cpu ipv6**
  - This feature adds support for traffic policy on IPv6 traffic destined to the CPU.

- **traffic-policy port ipv4**
  - This feature adds support for traffic policy on IPv4 traffic destined to egress a physical port.

- **traffic-policy port ipv6**
  - This feature adds support for traffic policy on IPv6 traffic destined to egress a physical port.

- **traffic-policy port ipv4 egress**
  - This feature adds support for traffic policy on IPv4 traffic egressing a physical port.
  - For details on how to use egress traffic-policy refer to the following TOI (/en/support/toi/eos-4-26-2f/14824-support-for-traffic-policy-on-egress-interfaces)

- **traffic-policy port ipv6 egress**
  - This feature adds support for traffic policy on IPv6 traffic egressing a physical port.
  - For details on how to use egress traffic-policy refer to the following TOI (/en/support/toi/eos-4-26-2f/14824-support-for-traffic-policy-on-egress-interfaces)

Sampled Flow Tracking IPv4 Hardware Offload

IP Flow Information Export (IPFIX) is a protocol used to export IP flow information from network elements. The hardware offload feature maintains the flow cache in hardware whilst also offloading CPU intensive tasks like packet parsing and counting packets and bytes for flows to the hardware. Please refer to the feature TOI (/en/support/toi/eos-4-25-2f/14697-sampled-flow-tracking-ipv4-hardware-offload) for more information.

- **flow tracking sampled ipv4**
  - This feature enables sampled flow tracking hardware offload for IPv4 traffic

Postcard Telemetry

The postcard telemetry feature is used to gather per flow telemetry information like path and per hop latency. Postcard telemetry samples a flow at every switch, aggregates them, and sends the samples to a collector with path and latency information using GRE encapsulation. Please refer to the feature TOI (/en/support/toi/eos-4-26-0f/14729-postcard-telemetry) for more information.

- **telemetry postcard policy ipv4**
  - This feature adds support for postcard telemetry on bridged and routed IPv4 traffic.

- **telemetry postcard policy ipv6**
  - This feature adds support for postcard telemetry on bridged and routed IPv6 traffic.

VRF Selection

VRF Selection allows you to assign a packet to a VRF based on a VRF selection policy that contains match rules with flexible criteria (e.g., DSCP, IP Protocol). The related VRF fallback feature is an extension of these policies which allows users to optionally specify a “fallback” VRF for each VRF. Please refer to the feature TOI (/en/support/toi/eos-4-26-2f/14810-vrf-selection-and-fallback) for more information.

- **vrf selection**
  - This feature adds support for VRF selection policies and VRF fallback

Miscellaneous features

- **cbf**

    - The ***cbf*** feature enables Class Based Forwarding. Class Based Forwarding (CBF) is a means for steering IP traffic into an Segment Routing (SR) Policy based on the ingress DSCP values.

- **cfm**
    - The feature enables trapping of CFM packets like CCM, DMM, LM by UP MEP

- **flex-route**
    - The feature ***flexroute*** gives users the option to optimize handling of routes with certain prefix lengths based on their network requirements and the scale of their routing table. Additional information about configuring flexroute can be found in the Configurations and Optimizations for Internet Edge Routing (https://arista.my.site.com/AristaCommunity/s/article/inet-edge-config) and DCS-7280SR2K with FLEXROUTE – A real world use case (https://arista.my.site.com/AristaCommunity/s/article/dcs-7280sr2k-with-flexroute-a-real-world-use-case) TOIs.

- **l2-protocol forwarding**
    - The ***l2-protocol forwarding*** feature enables selective forwarding of certain L2 protocol packets (VLAN tagged/ untagged/ both).
    - For details on how to use l2-protocol forwarding, please refer to the following TOI (/en/support/toi/eos-4-21-0f/14053-l2-protocol-forwarding).

- **tcp-mss-ceiling ip**
    - The ***tcp-mss-ceiling ip*** feature enables overwriting of the MSS value in TCP SYN packets prior to encapsulation if the packet MSS exceeds a certain value.

- **interface-policing**
    - This feature enables applying policiers to an interface to police the individual traffic coming through and/or the accumulated stream of traffic from groups of interfaces
    - On the R and R2 series of switches, interface policing is only supported on subinterfaces

    - For details on how to use interface policing please refer to the following TOI (/en/support/toi/eos-4-25-1f/14655-interface-policing-on-7280r-r2-7500r-r2)

- **ptp e2etransparent one-step**
    - This feature is required to correctly handle PTP packets using UDP as its transit protocol when Precision Time Protocol (PTP) one-step end-to-end transparent mode is enabled.
    - For details on how to use PTP one-step end-to-end transparent mode please refer to the following TOI (/en/support/toi/eos-4-20-5f/14026-ieee1588-ptp-one-step-end-to-end-transparent-clock-new-ptp-mode-support).

- **vlan editing extended-64**
    - On the R and R2 series of switches, this feature enables support for double-tag rewrite operations for Flexible Interface Encapsulation (FlexEncap).
    - For details on how to use FlexEncap please refer to the following TOI (/en/support/toi/eos-4-24-2f/14551-flexible-interface-encapsulation-flexencap).

- **rfc2544**
    - This feature adds support for RFC2544 Initiator and Reflector support on R3.

- **mlag egress filter disabled**

- **unicast rpf lookup-vrf**
    - This feature adds support for Split VRF uRPF resolution.

Actions

Actions define what operation should be taken when the packet matches a TCAM entry. Multiple actions can be applied simultaneously to the packet. If multiple features are hit for the same packet then as long as the actions don't overlap, all actions are applied to the packet. If the features overlap, the feature with the higher priority takes effect. There is a limited amount of action space per TCAM bank. Double wide banks, that is 320-bit banks, have twice the action space as single wide 160-bit banks. The various TCAM actions require differing amounts of action space. If the amount of action bits required is greater than

the amount available an error message will be given and the profile will fail to be programmed. To fix this issue, remove one or more configured actions from the feature.

- **add-inner-vlan-tag**
- **add-vlan-tag**
  - The ***`add-vlan-tag`*** action adds a VLAN tag to the Ethernet frame.
- **copy-ttl**
  - When a bridged packet is routed by a redirection and the forwarding header is changed with the ***`set-fwd-header`*** action, the TTL of the packet is changed 255. This action updates the TTL to one less than the TTL on the ingress packet. This action is used by feature ***`flow`***. See the DirectFlow Output Nexthop Action (/en/support/toi/eos-4-23-0f/14348-directflow-output-nexthop-action) TOI for additional information on DirectFlow configuration.
- **count**
  - The ***`count`*** action counts packets that hit a particular rule. For some features counters must be enabled with the ***`hardware counter feature`*** command.
  - Field size: 18-bits
- **drop**
  - The ***`drop`*** action overrides the current forwarding decision and indicates to drop the packet.
  - Field size: 12-bits
- **log**
  - **Log**
- **mirror**
  - The ***`mirror`*** action specifies to make a copy of the packet and send it to another destination.
- **permit**
  - The ***`permit`*** action specifies that the packet should continue on with its current forwarding decision. The permit action may be used to override

a default drop action or individual drop action rules.

- **ptp-override**
  - The ***`ptp-override`*** action is used to override PTP classification on packets. This action is primarily used with the feature ***`tapagg port`***.

- **police-group**
  - The ***`police-group`*** action is used to allocate a policer that polices the aggregated traffic from multiple interfaces. This action is used by the feature ***`interface-policing`***

- **police-interface**
  - The ***`police-interface`*** action is used to allocate a dedicated policer to an interface. This action is used by the feature ***`interface-policing`***

- **redirect**
  - The ***`redirect`*** action overrides the current forwarding decision and specify a different destination.

- **redirect-to-cpu**
  - The ***`redirect-to-cpu`*** action overrides the current forwarding decision and sends the packet to the host processor.

- **redirect-to-vxlan**
  - The ***`redirect-to-vxlan`*** action redirects a packet to a vxlan interface. This action is used by feature ***`flow`***.

- **remove-header-bytes**
  - The ***`remove-header-bytes`*** action removes bytes from the start of the packet. This action is primarily used for feature macsec-proxy to support vxlan routing and bridging over macsec proxy ports.

- **sample**
  - **Add sth**

- **set-drop-precedence**
  - The ***`set-drop-precedence`*** action sets the packet's drop precedence. Drop precedence is used to give packets a higher or lower probability of getting dropped when congestion occurs.

- **set-dscp**

- The *set-dscp* action specifies to rewrite the DSCP field in the packet.

- **set-ecn**
  - The set-ecn action set the packets Explicit Congestion Notification (ECN). ECN is an extension to the Internet Protocol and to the Transmission Control Protocol which allows end-to-end notification of network congestion without dropping packets. ECN uses the two least significant (rightmost) bits of the DiffServ field in IPv4 or IPv6 header to encode four different codepoints:
    - 00: Non ECN-Capable Transport - Non-ECT
    - 01: ECN Capable Transport - ECT(1)
    - 10: ECN Capable Transport - ECT(0)
    - 11: Congestion Encountered - CE

- **set-fwd-header**
  - The *set-fwd-header* action sets the forwarding header to enable redirection of bridged packets. This action is used by feature *flow*. See the DirectFlow Output Nexthop Action (/en/support/toi/eos-4-23-0f/14348-directflow-output-nexthop-action) TOI for additional information on DirectFlow configuration.

- **set-policer**
  - The set-policer action specifies the policer to use for matched packets.

- **set-tc**
  - The set-tc action overrides the forwarding traffic class. The traffic class is used for CoS queue selection.

- **set-ttl-3b**
  - The *set-ttl-3b* action is used to set the lower 3-bits of TTL field to a value and the remaining TTL bits to zero. In other words, set the TTL to a value between 0 and 7 inclusive. The time to Live (TTL) field is used by routers to determine if the packet has traversed too many hops and should be dropped.

- **set-unshared-policer**
  - The *set-unshared-policer* action is used to set unshared policers. Unshared policers contribute to meters programmed individually for

each applied interface. This contrasts the standard ***set-policer*** action where interfaces in the same chip must ***share*** the same meter. The unshared policer index is a combination of ingress interface and TCAM results. Index MSB comes from the ingress port. Index LSB comes from the TCAM action. The ***set-unshared-policer*** action cannot be used in conjunction with the ***set-policer*** action. See the unshared policer TOI (/en/support/toi/eos-4-28-2f/16096-unshared-qos-policy-map-policers) for additional information.

- **snoop**
  - The ***snoop*** action specifies to snoop matching packets. The packet is forwarded based on the current forwarding decision and a copy is sent to the CPU.

- **timestamp**

Action Space

Chip Type / System	80-bit TCAM key	160-bit TCAM key	320-bit TCAM key
7500E, DCS-7280SE	20 bits	40 bits	80 bits
7500R, DCS-7280QR, DCS-7280QRA, DCS-7280CR	24 bits	48 bits	96 bits
7500R2, DCS-7280CR2, DCS-7280CR2A, DCS-7280CR2K, DCS-7280SR2, DCS-7280SR2M, DCS-7280SR2A, DCS-7280SR2K	24 bits	48 bits	96 bits
DCS-7020TR, DCS-7020TRA	24 bits	48 bits	96 bits

Chip Type / System	80-bit TCAM key	160-bit TCAM key	320-bit TCAM key
7500R3, DCS-7280PR3, DCS-7280PR3K, DCS-7280DR3, DCS-7280DR3K, DCS-7280CR3, DCS-7280CR3K, DCS-7280CR3MK	32 bits	64 bits	128 bits

Action-Priority

As mentioned in the Action section above, overlapping actions matching multiple TCAM features utilize a priority value to determine which feature action value to apply. A default priority is provided and is viewable when showing the TCAM profile details, but can be overridden by applying a configurable **action-priority** value.

Note that for the R3 series duplicate priority values can be assigned, and the priority is then ordered alphabetically on the internal feature name. For the R2 series, the profile will yield an error upon programming if a duplicate priority is used.

Packet Types

Packets are divided into the following types based on what type of headers are present, what type of forwarding is performed, etc. A given type of packet can result in up to 8 160-bit lookups. If a profile configures more than 8 160-bit lookups on a particular packet type the profile programming will fail. Many packet types are supersets of other packet types. Packet types are prioritized such that more specific packet types have precedence over less specific packet types. A packet type that matches a more specific packet type won't also match the features associated with a less specific packet type. Care must be given when changing the packet types associated with features. For example, for a given feature, if a key field is matching on IPv4 header fields and an IPv6 packet type is added to the feature, the TCAM behavior will not work as expected because the IPv6 packet does not have the required IPv4 fields.

- **non-ip forwarding bridged**
  - The packet type ***`non-ip forwarding bridged`*** matches bridged packets that are neither IPv4 nor IPv6.

Ethernet	Not IP

- **ipv4 forwarding bridged**
  - The packet type ***`ipv4 forwarding bridged`*** matches bridged IPv4 packets.

Ethernet	IPv4

- **ipv6 forwarding bridged**
  - The packet type ***`ipv6 forwarding bridged`*** matches bridged IPv4 packets.

Ethernet	IPv6

- **mpls eth forwarding bridged decap**
  - The packet type ***`mpls eth forwarding bridged decap`*** matches bridged MPLS packets that encapsulate Ethernet frames and the MPLS header is decapsulated.

Ingress Packet

Outer Ethernet	MPLS	Inner Ethernet

Egress Packet

Inner Ethernet

- **mpls forwarding bridged decap**
  - The packet type matches the same packet type as above, but this is programmed as a higher priority match and needs to be used when "***`feature evpn`***" is present in the profile

- **mpls ipv4 forwarding bridged**

- The packet type ***mpls ipv4 forwarding bridged*** matches bridged MPLS packets that encapsulate IPv4 packets.

Ethernet	MPLS	IPv4

- **mpls ipv6 forwarding bridged**
  - The packet type ***mpls ipv6 forwarding bridged*** matches bridged MPLS packets that encapsulate IPv6 packets.

Ethernet	MPLS	IPv6

- **mpls ipv4 ipv4 forwarding bridged**
  - The packet type ***mpls ipv4 ipv4 forwarding bridged*** matches bridged MPLS packets that encapsulate IPv4-in-IPv4 packets.

Ethernet	MPLS	IPv4	IPv4

- **mpls ipv4 ipv6 forwarding bridged**
  - The packet type ***mpls ipv4 ipv6 forwarding bridged*** matches bridged MPLS packets that encapsulate IPv6-in-IPv4 packets.

Ethernet	MPLS	IPv4	IPv6

- **mpls ipv6 ipv4 forwarding bridged**
  - The packet type ***mpls ipv6 ipv4 forwarding bridged*** matches bridged MPLS packets that encapsulate IPv4-in-IPv6 packets.

Ethernet	MPLS	IPv6	IPv4

- **mpls ipv6 ipv6 forwarding bridged**
  - The packet type ***mpls ipv6 ipv6 forwarding bridged*** matches bridged MPLS packets that encapsulate IPv6-in-IPv6 packets.

Ethernet	MPLS	IPv6	IPv6

- **ipv4 vxlan forwarding bridged decap**

- The packet type ***ipv4 vxlan forwarding bridged decap*** matches bridged IPv4 packets that encapsulate Vxlan and the outer Ethernet, IPv4, UDP, and VXLAN headers are decapsulated.

Ingress Packet

Outer Ethernet	Outer IPv4	Outer UDP	VXLAN	Inner Ethernet

Egress Packet

Inner Ethernet

- **ipv4 forwarding bridged sub-interface**
  - The packet type ***ipv4 forwarding bridged sub-interface*** matches bridged IPv4 packets that enter on a sub-interface.

Ethernet	IPv4

- **ipv6 forwarding bridged sub-interface**
  - The packet type ***ipv6 forwarding bridged sub-interface*** matches bridged IPv6 packets that enter on a sub-interface.

Ethernet	IPv6

- **non-ip forwarding bridged sub-interface**
  - The packet type ***non-ip forwarding bridged sub-interface*** matches bridged frames that do not encapsulate IPv4 or IPv6 packets that enter on a sub-interface.

Ethernet	Not IP

- **ipv4 forwarding routed**
  - The packet type ***ipv4 forwarding routed*** matches routed IPv4 packets.

Ethernet	IPv4

- **ipv4 non-vxlan forwarding routed decap**

- The packet type ***ipv4 non-vxlan forwarding routed decap*** matches routed IPv4 packets that get tunnel terminated where the tunnel type is not vxlan (for example, GRE).

- **ipv4 non-vxlan forwarding routed multicast decap**
  - The packet type ***ipv4 non-vxlan forwarding routed multicast decap*** matches routed IPv4 packets that get tunnel terminated where the tunnel type is not vxlan and inner destination ip address is an IPv4 multicast address.

- **ipv4 vxlan eth ipv4 forwarding routed multicast decap**
  - The packet type ***ipv4 vxlan eth ipv4 forwarding multicast routed decap*** matches routed IPv4 multicast packets encapsulating UDP, VXLAN, Ethernet, and IPv4 headers and the outer Ethernet, IPv4, UDP, and VXLAN headers are decapsulated.

- **ipv4 vxlan eth ipv4 forwarding routed decap**
  - The packet type ***ipv4 vxlan eth ipv4 forwarding routed decap*** matches routed IPv4 packets encapsulating UDP, VXLAN, Ethernet, and IPv4 headers and the outer Ethernet, IPv4, UDP, and VXLAN headers are decapsulated.

  Ingress Packet

Outer Ethernet	Outer IPv4	Outer UDP	VXLAN	Inner Ethernet	Inner IPv4

  Egress Packet

New Ethernet	Inner IPv4

- **ipv4 vxlan eth ipv6 forwarding routed decap**
  - The packet type ***ipv4 vxlan eth ipv6 forwarding routed decap*** matches VXLAN IPv4 underlay packets with IPv6 overlay, that are decapsulated and routed on the inner IPv6 header

  Ingress Packet

Outer Ethernet	Outer IPv4	Outer UDP	VXLAN	Inner Ethernet	Inner IPv6

  Egress Packet

New Ethernet	Inner IPv6

- **ipv4 forwarding routed multicast**
  - The packet type ***`ipv4 forwarding routed multicast`*** matches routed multicast IPv4 packets.

Ethernet	Multicast IPv4

- **ipv4 forwarding routed multicast disabled**
  - The packet type ***`ipv4 forwarding routed multicast disabled`*** will avoid matching on routed multicast IPv4 packets.
  - This packet type is only applicable to feature `acl port ip egress`**.**

Ethernet	Multicast IPv4

- **ipv6 forwarding routed multicast**
  - The packet type ***`ipv6 forwarding routed multicast`*** matches routed multicast IPv6 packets.

Ethernet	Multicast IPv6

- **ipv6 forwarding routed**
  - The packet type ***`ipv6 forwarding routed`*** matches routed IPv6 packets.

Ethernet	IPv6

- **ipv6 forwarding routed decap**
  - The packet type ***`ipv6 forwarding routed decap`*** matches routed IPv6 packets after tunnel termination.
- **ipv6 ipv4 forwarding routed decap**

- The packet type ***ipv6 ipv4 forwarding routed decap*** matches routed IPv4 over IPv6 packets and the outer IPv6 header is decapsulated.

Outer Ethernet	Outer IPv6	Inner IPv4

- **ipv6 ipv6 forwarding routed decap**
  - The packet type ***ipv6 ipv6 forwarding routed decap*** matches routed IPv6 over IPv6 packets and the outer IPv6 header is decapsulated.

  Ingress Packet

Outer Ethernet	Outer IPv6	Inner IPv6

  Egress Packet

New Ethernet	Inner IPv6

- **Ipv6 vxlan forwarding bridged decap**
  - The packet type ***ipv6 vxlan forwarding bridged decap*** matches bridged IPv6 packets that encapsulate Vxlan and the outer Ethernet, IPv6, UDP, and VXLAN headers are decapsulated.

  Ingress Packet

Outer Ethernet	Outer IPv6	Outer UDP	VXLAN	Inner Ethernet

  Egress Packet

Inner Ethernet

- **ipv6 vxlan eth ipv6 forwarding routed decap**
  - The packet type ***ipv6 vxlan eth ipv6 forwarding routed decap*** matches VXLAN IPv6 underlay packets with IPv6 overlay, that are decapsulated and routed on the inner IPv46 header

  Ingress Packet

Outer Ethernet	Outer IPv6	Outer UDP	VXLAN	Inner Ethernet	Inner IPv6

Egress Packet

New Ethernet	Inner IPv6

- **ipv6 vxlan eth ipv4 forwarding routed decap**
  - The packet type ***`ipv6 vxlan eth ipv4 forwarding routed decap`*** matches VXLAN IPv6 underlay packets with IPv4 overlay, that are decapsulated and routed on the inner IPv4 header

Ingress Packet

Outer Ethernet	Outer IPv6	Outer UDP	VXLAN	Inner Ethernet	Inner IPv4

Egress Packet

New Ethernet	Inner IPv4

- **mpls non-ip forwarding mpls**
  - The packet type ***`mpls non-ip forwarding mpls`*** matches MPLS label switched packets that do not encapsulate IPv4 or IPv6 packets.

Ethernet	MPLS	Not IP

- **mpls ipv4 forwarding mpls**
  - The packet type ***`mpls ip forwarding mpls`*** matches MPLS label switched packets that encapsulate an IPv4 packet.

Ethernet	MPLS	IPv4

- **mpls ipv6 forwarding mpls**
  - The packet type ***`mpls ipv6 forwarding mpls`*** matches MPLS label switched packets that encapsulate an IPv6 packet.

Ethernet	MPLS	IPv6

- **ipv4 mpls ipv4 forwarding mpls decap**
  - The packet type ***`ipv4 mpls ipv4 forwarding mpls decap`*** matches IPv4 packets which encapsulate GRE, MPLS, and IPv4 headers. The outer IPv4 and GRE headers are decapsulated, the ethernet header is changed according to forwarding semantics and the MPLS header is MPLS label switched. See the MPLS (/en/um-eos/eos-multiprotocol-label-switching-mpls) section in the user manual for more details.

Ingress Packet

Ethernet	Outer IPv4	GRE	MPLS	Inner IPv4

Egress Packet

New Ethernet	New MPLS	Inner IPv4

- **ipv4 mpls ipv6 forwarding mpls decap**
  - The packet type ***`ipv4 mpls ipv6 forwarding mpls decap`*** matches IPv4 packets which encapsulate GRE over MPLS over IPv6. The outer IPv4 and GRE headers are decapsulated, the ethernet header is changed according to forwarding semantics and the MPLS header is MPLS label switched.

Ingress Packet

Ethernet	Outer IPv4	GRE	MPLS	Inner IPv6

Egress Packet

New Ethernet	New MPLS	Inner IPv6

- **ipv4 mpls ipv4 forwarding routed**
  - The packet type ***`ipv4 mpls ipv4 forwarding routed`*** matches transit IPv4 packets which encapsulate GRE, MPLS, and IPv4 headers.

Ethernet	Outer IPv4	GRE	MPLS	Inner IPv4

- **ipv4 mpls ipv6 forwarding routed**
  - The packet type ***`ipv4 mpls ipv6 forwarding routed`*** matches transit IPv4 packets which encapsulate GRE, MPLS, and IPv6 headers.

Ethernet	Outer IPv4	GRE	MPLS	Inner IPv6

- **mpls ipv4 forwarding routed decap**
  - The packet type ***`mpls ipv4 forwarding routed decap`*** matches MPLS packets which encapsulate IPv4. The MPLS header is popped, the packet is routed on the IPv4 header and the ethernet header is changed according to the routing decision

Ingress Header

Ethernet	MPLS	IPv4

Egress Header

New Ethernet	IPv4

- **mpls ipv6 forwarding routed decap**
  - The packet type ***`mpls ipv6 forwarding routed decap`*** matches MPLS packets which encapsulate IPv6. The MPLS header is popped, the packet is routed based on the IPv6 header and the ethernet header is changed according to the routing decision.

Ingress Header

Ethernet	MPLS	IPv6

Egress Header

New Ethernet	IPv6

- **ipv4 vxlan forwarding bridged**
  - The packet is bridged on the outer-ethernet header

Ingress/Egress Header

Outer-Ethernet	IPv4	UDP	VXLAN	Inner-Ethernet

- **ipv4 vxlan forwarding routed**
  - The packet is routed on the outer-IPv4 header

Ingress Header

Outer-Ethernet	IPv4	UDP	VXLAN	Inner-Ethernet

Egress Header

Outer-Ethernet (rewritten)	IPv4 (TTL/checksum update)	UDP	VXLAN	Inner-Ethernet

- **ipv4 eth ipv4 forwarding bridged**
  - The packet type ***`ipv4 eth ipv4 forwarding bridged`*** matches IPv4 packets which encapsulate GRE, inner Ethernet and inner IPv4 headers.

  Ingress/Egress Header

Outer-Ethernet	IPv4	Inner-Ethernet	IPv4

- **ipv4 gtpv1 ipv4 forwarding bridged**
  - The packet type ***`ipv4 gtpv1 ipv4 forwarding bridged`*** matches IPv4 packets which encapsulate GTPv1 protocol, and inner IPv4 headers.

  Ingress/Egress Header

Ethernet	IPv4	GTPv1	IPv4

- **ipv4 gtpv1 ipv6 forwarding bridged**
  - The packet type ***`ipv4 gtpv1 ipv6 forwarding bridged`*** matches IPv4 packets which encapsulate GTPv1 protocol, and inner IPv6 headers.

  Ingress/Egress Header

Ethernet	IPv4	GTPv1	IPv6

- **ipv6 gtpv1 ipv4 forwarding bridged**
  - The packet type ***`ipv6 gtpv1 ipv4 forwarding bridged`*** matches IPv6 packets which encapsulate GTPv1 protocol, and inner IPv4 headers.

  Ingress/Egress Header

Ethernet	IPv6	GTPv1	IPv4

- **ipv6 gtpv1 ipv6 forwarding bridged**

- The packet type ***ipv6 gtpv1 ipv6 forwarding bridged*** matches IPv6 packets which encapsulate GTPv1 protocol, and inner IPv6 headers.

Ingress/Egress Header

Ethernet	IPv6	GTPv1	IPv6

Key Fields

Key fields extract data from packet headers or results of previously computed values in the forwarding pipeline. The key fields are matched in the TCAM against programmed values. For example, a given feature may specify the ***src-ip*** key field as one of its key fields. The feature might program the TCAM to take a different action on matching a certain IPv4 source IP address. In the TCAM, all key fields are matched simultaneously. However, any given key field can be masked out such that it doesn't participate in the match decision.

Ethernet Fields

The following fields are extracted from the Ethernet header.

- **dst-mac** (MAC destination address)
  - Field Size: 48-bits, This field is broken into a 32-bit chunk and a 16-bit chunk in the hardware.
- **src-mac** (MAC source address)
  - Field Size: 48-bits, This field is broken into a 32-bit chunk and a 16-bit chunk in the hardware.
- **outer-vlan-id** (Outer VLAN ID)
  - This is the vlan id in the first / outer 802.1Q header. If there is only one 802.1Q header, this is that vlan id in that header.
  - Field Size: 12-bits
- **inner-cos** (Inner Ethernet Class of Service)
  - When the Ethernet frame has two 802.1Q headers, this is the Priority code point (PCP) field in the second / inner 802.1Q header.
  - Field Size: 3 bits

- **inner-vlan-id** (Inner VLAN ID)
  - When the Ethernet frame has two 802.1Q headers, this is the vlan id in the second / inner 802.1Q header.
  - Field Size: 12-bits
- **cos** (Ethernet Class of Service)
  - This is the Priority code point (PCP) field in the 802.1Q header.
  - Field Size: 3-bits
- **ether-type** (Ethertype protocol value)
  - **Field Size: 16-bits**

Ethernet Internal Fields

The following fields are based on internal hardware state. They are based on information obtained from packet headers and previous decisions in the forwarding pipeline.

- **vlan-tag-format** (Ethernet packet VLAN tag format)
  - Field Size: 4-bits
- **vlan** (Forwarding VLAN ID)
  - Field Size: 12-bits

Encapsulated Ethernet Fields

- tunnelled-1st-tpid-exists (Inner Ethernet 1st TPID matches 0x8100, 0x9100, 0x88a8, 0x88e7)
  - Field Size: 1-bit
- tunnelled-2nd-tpid-exists (Inner Ethernet 2nd TPID matches 0x8100, 0x9100, 0x88a8, 0x88e7)
  - Field Size: 1-bit
- tunnelled-1st-vlan-id (Inner Ethernet Forwarding VLAN ID from the 1st TPID)
  - Field Size: 12-bits
- tunnelled-2nd-vlan-id (Inner Ethernet Forwarding VLAN ID from the 2nd TPID)

- Field Size: 12-bits

IPv4 Fields

The following fields are extracted from the IPv4 header.

- **dscp** (IPv4 DSCP value)
  - Field Size: 8-bits, even though DSCP field in the IPv4 header is 6-bits, the 8-bit DS field is copied

- **dst-ip** (IPv4 destination address)
  - Field Size: 32-bits

- **dst-ip-high** (IPv4 destination address (upper 16 bits))
  - Field Size: 16-bits

- **dst-ip-low** (IPv4 destination address (lower 16 bits))
  - Field Size: 16-bits

- **src-ip** (IPv4 source address)
  - Field Size: 32-bits

- **src-ip-high** (IPv4 source address (upper 16 bits))
  - Field Size: 16-bits

- **src-ip-low** (IPv4 source address (lower 16 bits))
  - Field Size: 16-bits

- **ip-frag** (Non-head packet fragment)
  - Field Size: 1-bit

- **ip-fragment-offset** (packet fragment offset)
  - The key field ***`ip-fragment-offset`*** is first supported in 4.22.1F. It is supported by R3 line cards and fixed systems. It will be supported by other card types in a future release.

  - Field Size: 13-bits

- **ip-length** (IP packet length)
  - Field Size: 16-bits

- **ip-protocol** (IP protocol number)

- Field Size: 8-bits

- **ipv4-id** (IPv4 identification)
  - Field Size: 16-bits

- **tos** (Type of service)
  - Field Size: 8-bits

- **ttl** (Time-To-Live (TTL) value)
  - Field Size: 8-bits

IPv4 Internal Fields

The following fields are based on internal hardware state. They are based on information obtained from packet headers and previous decisions in the forwarding pipeline.

- **dst-ip-label** (IPv4 destination transformed prefix)
  - The key field ***`dst-ip-label`*** is first supported in 4.22.1F. It is supported by R3 line cards and fixed systems.
  - Field Size: 16-bits

- **src-ip-label** (IPv4 source transformed prefix)
  - The key field ***`src-ip-label`*** is first supported in 4.22.1F. It is supported by R3 line cards and fixed systems.
  - Field Size: 16-bits

- **dst-ip-lpm-label** (IPv4 destination transformed prefix for longest prefix matching)
  - The key field ***`dst-lpm-label`*** is first supported in Tokyo. It is supported by R3 line cards and fixed systems.
  - Field Size: 16-bits

- **src-ip-lpm-label** (IPv4 source transformed prefix for longest prefix matching)
  - The key field ***`src-ip-lpm-label`*** is first supported in Tokyo. It is supported by R3 line cards and fixed systems.
  - Field Size: 16-bits

- **ip-tunnel-hit** (IP tunnel lookup hit (pipeline field))
- **ip-type** (IP type parsed from Ethertype Ethernet field)
  - Field Size: 5-bits on R3 line cards and R3 fixed systems.
  - Field Size: 4-bits on pre-R3 line cards and pre-R3 fixed systems.

Encapsulated IPv4 Fields

The following fields are extracted from an encapsulated IPv4 header.

- **inner-dst-ip** (Inner IP header destination address)
  - Field Size: 32-bits
- **inner-dst-ip-high** (Inner IP header destination address (upper 16 bits))
  - Field Size: 16-bits
- **inner-dst-ip-low** (Inner IP header destination address (lower 16 bits))
  - Field Size: 16-bits
- **inner-src-ip** (Inner IP header source address)
  - Field Size: 32-bits
- **inner-src-ip-high** (Inner IP header source address (upper 16 bits))
  - Field Size: 16-bits
- **inner-src-ip-low** (Inner IP header source address (lower 16 bits))
  - Field Size: 16-bits
- **inner-ip-frag** (Inner non-head packet fragment)
  - Field Size: 1-bit
- **inner-ip-protocol** (Inner IP header protocol number)
  - Field Size: 8-bits
- **inner-tos** (Inner IP header TOS value)
  - Field Size: 8-bits
- **inner-ttl** (Inner IP header TTL value)
  - Field Size: 8-bits

IPv6 Fields

The following fields are extracted from the IPv6 header.

- **dst-ipv6** (IPv6 destination address)
  - Field Size: 128-bits, This field is broken into four 32-bit chunks in the hardware.

- **dst-ipv6-high** (IPv6 destination address (upper 64 bits))
  - Field Size: 64-bits, This field is broken into two 32-bit chunks in the hardware.

- **dst-ipv6-low**, IPv6 destination address (lower 64 bits))
  - Field Size: 64-bits, This field is broken into two 32-bit chunks in the hardware.

- **dst-ipv6-bits-96-111**
  - Field size : 16 bits. This field matches bits 96-111 of IPv6 destination address.

- **src-ipv6** (IPv6 source address)
  - Field Size: 128-bits, This field is broken into four 32-bit chunks in the hardware.

- **src-ipv6-high** (IPv6 source address (upper 64 bits))
  - Field Size: 64-bits, This field is broken into two 32-bit chunks in the hardware.

- **src-ipv6-low** (IPv6 source address (lower 64 bits))
  - Field Size: 64-bits, This field is broken into two 32-bit chunks in the hardware.

- **src-ipv6-low-high-32b** (IPv6 source address (second lowest 32 bits))
  - Field Size: 32-bits

- **src-ipv6-low-low-32b** (IPv6 source address (lowest 32 bits))
  - Field Size: 32-bits

- **src-ipv6-bits-96-111**

- **ipv6-flow-label (IPv6 flow label)**
  - Field Size: 20-bits

- **ipv6-length** (IPv6 packet length)
  - Field Size: 16-bits
- **ipv6-next-header** (IPv6 next header/IP protocol type)
  - Field Size: 8-bits
- **ipv6-traffic-class** (IPv6 traffic class)
  - Field Size: 8-bits
- **hop-limit** (IPv6 hop limit)
  - The key field ***`hop-limit`*** is first supported in 4.24.0F. It is supported by the R3 line cards and fixed systems. It will be supported on additional line cards in a future release.
  - Field Size: 8-bits

IPv6 Internal Fields

The following fields are based on internal hardware state. The are based on information obtained from packet headers and previous decisions in the forwarding pipeline.

- **dst-ipv6-label** (IPv6 destination transformed prefix)
  - The key field ***`dst-ipv6-label`*** is first supported in 4.24.0F. It is supported by the R3 line cards and fixed systems.
  - Field Size: 16-bits
- **src-ipv6-label** (IPv6 source transformed prefix)
  - The key field ***`src-ipv6-label`*** is first supported in 4.24.0F. It is supported by the R3 line cards and fixed systems.
  - Field Size: 16-bits
- **dst-ipv6-lpm-label** (IPv6 destination transformed prefix for longest prefix matching)
  - The key field ***`dst-ipv6-lpm-label`*** is first supported in Tokyo. It is supported by the R3 line cards and fixed systems.
  - Field Size: 16-bits
- **src-ipv6-lpm-label** (IPv6 source transformed prefix for longest prefix matching)

    - The key field ***src-ipv6-lpm-label*** is first supported in Tokyo. It is supported by the R3 line cards and fixed systems.
    - Field Size: 16-bits
- **parsed-next-header (IPv6 next header/IP protocol type, skipping over at most one IPv6 extension header)**
    - Field Size: 5-bits

Encapsulated IPv6 Fields

The following fields are extracted from an encapsulated IPv6 header.

- **inner-dst-ipv6-high-high-32b** (Inner IPv6 destination address (highest 32 bits))
    - Field Size: 32-bits
- **inner-dst-ipv6-high-low-32b** (Inner IPv6 destination address (third lowest 32 bits))
    - Field Size: 32-bits
- **inner-dst-ipv6-low-high-32b** (Inner IPv6 destination address (second lowest 32 bits))
    - Field Size: 32-bits
- **inner-dst-ipv6-low-low-32b** (Inner IPv6 destination address (lowest 32 bits))
    - Field Size: 32-bits
- **inner-src-ipv6-high-high-32b** (Inner IPv6 source address (highest 32 bits))
    - Field Size: 32-bits
- **inner-src-ipv6-high-low-32b** (Inner IPv6 source address (third lowest 32 bits))
    - Field Size: 32-bits
- **inner-src-ipv6-low-high-32b** (Inner IPv6 source address (second lowest 32 bits))
    - Field Size: 32-bits

- **inner-src-ipv6-low-low-32b** (Inner IPv6 source address (lowest 32 bits))
  - Field Size: 32-bits
- **inner-ipv6-next-header** (Inner IPv6 next header protocol number)
  - Field Size: 8-bits

MPLS Fields

The following fields are extracted from the MPLS header.

- **mpls-forwarding-label-action** (MPLS forwarding label action)
- **mpls-forwarding-label-ttl** (TTL routed field of MPLS forwarding label)
- **mpls-inner-ip-tos** (Inner IP header ToS field of the MPLS packet)
  - Field Size: 16-bits
- **mpls-label1-lower-24b** (Lower 24-bits of the first MPLS label)
  - Field Size: 24-bits
- **mpls-label2-lower-24b** (Lower 24-bits of the second MPLS label)
  - Field Size: 24-bits
- **mpls-label3-lower-24b** (Lower 24-bits of the third MPLS label)
  - Field Size: 24-bits
- **mpls-traffic-class** (MPLS traffic class)
  - Field Size: 3-bits
- **propagated-ttl** (Propagated TTL as determined by uniform/pipe mode setting)
  - Field Size: 8-bits

Vxlan Fields

- **vxlan-inner-etype** (Ethertype protocol value of VXLAN inner Ethernet header)
  - Field Size: 4-bits
- **vxlan-inner-ip-options** (IP options of VXLAN inner IP header)
  - Field Size: 1-bit

- **vxlan-inner-ip-ttl** (TTL (Time-To-Live) value of VXLAN inner IP header)
  - Field Size: 8-bits
- **vxlan-inner-ip-udp-dport** (UDP destination port of the inner IPv4/v6 header in a VXLAN packet)
  - Field size: 16-bits
- **vxlan-udp-port** (UDP port in VXLAN header)
  - Field size: 16-bits
- **vxlan-inner-icmpv6-type (Inner ICMPV6 packet type of a VXLAN encapsulated ICMPv6 packet)**
  - Field size: 8-bits

GRE Fields

- **gre-flags** (First 16b of the GRE header)
  - Field size: 16-bits
  - For use in tapagg!tunnel!termination feature
- **gre-protocol** (Protocol field of the GRE header)
  - Field size: 16-bits
  - For use in tapagg!tunnel!termination feature

Layer 4 Fields

The following fields are extracted from the TCP, UDP, and ICMP headers.

- **l4-dst-port** (Destination port number)
  - Field Size: 16-bits
- **l4-ops** (L4 port range)
  - Field Size: 24-bits
- **l4-ops-18b** (First 18-bits of L4 port range)
  - Field Size: 18-bits
- **l4-ops-2b** (First 2-bits of L4 port range)
  - Field Size: 2-bits

- **l4-ops-3b** (First 3-bits of L4 port range)
  - Field Size: 3-bits
- **l4-ops-6b** (First 6-bits of L4 port range)
  - The key field l4-ops-6b is first supported in 4.29.2F. It is supported by the R3 Systems.
  - Field Size: 6-bits
- **l4-ops-7b** (First 7-bits of L4 port range)
  - Field Size: 7-bits
- **l4-src-port** (Source port number)
  - Field Size: 16-bits
- **tcp-control** (TCP control flags)
  - Field Size: 6-bits
- **tcp-header-length** (TCP header length)
  - Field size:
- **tcp-first-option-kind** (TCP first option kind)
  - Field size:
- **tcp-checksum** (TCP checksum)
  - Field Size: 16-bits
- **icmp-type-code** (ICMP type and code fields)
  - The icmp-type-code is taken from the type and code fields in the ICMP header.
  - Field Size: 16-bits
- **udp-checksum** (UDP checksum)
  - Field Size: 16-bits

Layer 4 Internal Fields

The following fields are based on internal hardware state. They are based on information obtained from packet headers and previous decisions in the forwarding pipeline.

- **l4-src-port-label** (Source transformed port number)
  - The key field ***`l4-src-port-label`*** is first supported in 4.24.0F. It is supported by the R3 line cards and fixed systems.
  - Field Size: 16-bits
- **l4-dst-port-label** (Destination transformed port number)
  - The key field ***`l4-dst-port-label`*** is first supported in 4.24.0F. It is supported by the R3 line cards and fixed systems.
  - Field Size: 16-bits

Inner Layer 4 Fields

The following fields are extracted from encapsulated L4 headers.

- **inner-l4-dst-port** (Inner IP header destination port)
  - Field Size: 16-bits
- **inner-l4-src-port** (Inner IP header source port)
  - Field Size: 16-bits
- **inner-tcp-control** (Inner TCP control flags)
  - Field Size: 6-bits

Aegis-specific Fields

The following fields are extracted from...

- **src-ip-label** (Source transformed prefix label)
  - Field Size: 16-bits
- **dst-ip-label** (Destination transformed prefix label)
  - Field Size: 16-bits
- **gtpv1-teid-label**
- **inner-src-ip-label** (Inner source transformed prefix label for encapsulated packet)
  - Field Size: 16-bits

- **inner-dst-ip-label** (Inner destination transformed prefix label for encapsulated packet)
  - Field Size: 16-bits
- **l4-src-port-label** (Source transformed port number)
  - Field Size: 16-bits
- **l4-dst-port-label** (Destination transformed port number)
  - Field Size: 16-bits

Miscellaneous Fields

- **acl-label**
- **dst-port** (Destination module port number)
  - The dst-port field matches on the destination module ID and port. Do not confuse this with the l4-dst-port which matches on the TCP/UDP header port.
  - Field Size: 16-bits
- **esi-label** (ESI (Ethernet Segment Identifier) label)
  - Field Size: 20-bits
- **erspan-sub-header**
- **forwarding-type**, Packet forwarding type (bridged or routed)
- **in-port** (Ingress interface)
  - Field Size: 8-bits
- **ip-version-and-tc** (IP version and ToS/traffic-class)
  - Field Size: 16-bits
- **qos-policy-tc** (Traffic Class set by a qos policy map)
  - Field Size: 4-bits
- **resolved-tc** (Resolved Traffic Class including port configuration)
  - Untrusted ports use the port TC instead of the packet TC; trusted ports use the mapped TC default 0->1, 1->0, 2->2, 3->3, ... 7->7
  - Field Size: 3-bits

- **traffic-policy-tc** (Traffic Class set by a traffic policy)
  - **Field Size: 4-bits**
- **vrf** (Virtual Routing and Forwarding)
  - Field Size: 16-bits

User-Defined Fields (UDF) Fields

The purpose of the User-Defined Fields feature is to permit or deny packets based on custom offset pattern matching. User-Defined Fields are defined as part of an access-list filter and are comprised of an offset, length, pattern match and mask. This describes a single portion of any incoming packet to match the provided value upon. For more information about User-Defined Fields please refer to the Port ACLs with User-Defined Fields (/en/support/toi/eos-4-23-1f/14399-port-acls-with-user-defined-fields) TOI.

- **udf-16b-1** (16-bit UDF qualifier)
  - Field Size: 16-bits
- **udf-16b-2** (16-bit UDF qualifier)
  - Field Size: 16-bits
- **udf-16b-3** (16-bit UDF qualifier)
  - Field Size: 16-bits
- **udf-16b-4** (16-bit UDF qualifier)
  - Field Size: 16-bits
- **udf-16b-5** (16-bit UDF qualifier)
  - Field Size: 16-bits
- **udf-16b-6** (16-bit UDF qualifier)
  - Field Size: 16-bits
- **udf-32b-1** (32-bit UDF qualifier)
  - Field Size: 32-bits
- **udf-32b-2** (32-bit UDF qualifier)
  - Field Size: 32-bits

- **udf-32b-3** (32-bit UDF qualifier)
  - Field Size: 32-bits
- **udf-32b-4** (32-bit UDF qualifier)
  - Field Size: 32-bits
- **udf-32b-5** (32-bit UDF qualifier)
  - Field Size: 32-bits
- **udf-32b-6** (32-bit UDF qualifier)
  - Field Size: 32-bits
- **udf-32b-7** (32-bit UDF qualifier)
  - Field Size: 32-bits

Other TCAM Profile Attributes

- **prefix label size**
- **prefix lpm label size**
- **policy policer unshared action size**
- **policy policer unshared qualifier size**

Syslogs

Errors (TCAM-3-PROGRAMMING_FAILURE):

There are also several TCAM error messages that will fail the profile programming, and use the **TCAM-3-PROGRAMMING_FAILURE** syslog.

These messages follow a standard template:
"***Failed to program configured TCAM profile <profile> on <linecard#|fixedsystem> because <error message>***"

The following ***<error messages>*** currently exist:

- "***no action is configured in feature <feature>***"
  - There are not any actions configured for this feature. At least one action is required. Actions are required for most features. Actions

specify what will be done on a packet match in the TCAM. When no actions are specified, that TCAM feature configuration is essentially meaningless and we flag it as an error. However, there are exceptions. If the feature does not require any tcam banks, the feature doesn't need an action.

- "***no key field is configured in feature <feature>***"
  - There are not any key fields configured for this feature. At least one key field is required. Key fields specify what fields in the packet or hardware forwarding pipeline are compared. Not specifying any key fields is a misconfiguration. However, there are exceptions. If the feature does not require any tcam banks, the feature doesn't need a key field.

- "***no packet type is configured in feature <feature>***"
  - No packet type is configured and a packet type to match on is required. Packet types configured on a feature specify what types of packets will be matched against in the TCAM. If no packet types are specified, no packets will be matched against that TCAM and is considered a misconfiguration. However, there are exceptions. If the feature does not require any tcam banks, the feature doesn't need any packet type configuration.

- "***the key size of feature <feature> exceeds the configured key size limit***"
  - The key size is insufficient to program the feature. The key size limit either needs to be increased or key fields need to be removed. The key size limit configuration is used to ensure that features' key size does not inadvertently exceed a set size limit. In the TCAM hardware, there are three supported key sizes, 80, 160, and 320-bit. Expanding the key size from 80 to 160 or 160 to 320, requires twice the amount of TCAM bank resources. However, the key size limit is a pure configuration construct. If a profile has the resources to expand the key size limit, it is fine to increase it. If the feature cannot be programmed with a larger key size, then unused key fields need to be removed. If all key fields are required, additional trade-offs need to be made.

- "***the action size of feature <feature> exceeds the hardware platform limit***"

    - The size of the configured actions exceeds the hardware capability. One or more actions need to be removed to allow programming.
- "***the key size of feature <feature> is larger than the configured primary key size limit of feature sharing the same TCAM bank***"
    - The key size must match the key size limit of the shared feature bank.
- "***action <action> is not supported on this hardware platform***"
    - The platform does not support the action configured.
- "***feature <feature> is not supported on this hardware platform***"
    - The platform does not support the feature configured.
- "***qualifier <qualifier> is not supported on this hardware platform***"
    - The platform does not support the qualifier configured.
- "***packet type <packet type> is not supported on this hardware platform***"
    - The platform does not support the packet type configured.
- "***Resources for extracting key-fields for feature <feature> exceeded on this hardware platform***"
    - Key field extraction resources exceeded. Reduce key fields.
- "***dynamic port qualifier size is not supported for feature <feature>***"
    - Dynamic sizing of the port qualifier is not supported by this feature.
- "***dynamic port qualifier size is not supported on feature <feature> with AlgoMatch***"
    - Dynamic sizing of the port qualifier is not supported by this feature with AlgoMatch mechanism.
- "***The following features are mutually exclusive to feature <feature>:***"
    - The features conflict and cannot be programmed together.
- "***profile programming encounters internal failures***"
    - Unknown, general TCAM programming error.

Warnings:

There are also several TCAM warning messages that will not fail the profile programming, but should be reviewed to ensure configuration is what is actually desired.

These messages use the **TCAM-4-PROGRAMMING_WARNING** syslog and follow a standard template:
<warning message> in TCAM profile <profile> on <linecard#|fixedsystem>

The following ***<warning messages>*** currently exist:

- "***security ACL feature changes cannot be supported with AlgoMatch***"
  - The feature will be ignored on AlgoMatch switches and has no effect.
- "***feature <feature> is not configurable***"
  - The direct extracted DB changed and is not supported; it will have no effect.
- "***egress database feature changed***"
  - The egress DB changed and is not supported; it will have no effect.
- "***action <action> is not supported***"
  - The action configured is not supported on the chip type and has no effect.

Non-template warnings:

Some TCAM programming errors do not use the **t**emplate message but can also result in undesired behavior.

- **MATCH-6-RULE_FIELD_UNAVAILABLE:**
  - When an ACL rule is programmed with a missing key in the ACL configuration, the rule is programmed but without the missing key from the TCAM profile:
    MATCH-6-RULE_FIELD_UNAVAILABLE: The <field> field configured in the <rule> rule in <name> is not present in the TCAM key
- **TCAM-4-FEATURE_NOT_SUPPORTED:**

- If a feature is not supported by the TCAM profile, the following syslog is seen. For example, when a CPU traffic policy is applied and the underlying TCAM profile does not support the feature '***traffic-policy cpu***':
  %TCAM-4-FEATURE_NOT_SUPPORTED: The CPU Traffic Policy Matching on IPv4 Packets feature is not supported in the current TCAM profile programmed on Linecard4

- **TCAM_ACTION_NOT_AVAILABLE:**
  - When a TCAM profile feature setting is missing an action that is not present in the corresponding feature's definition, the feature will not work correctly:
    The <action> action required to support the <feature> feature is not present in the TCAM profile configured on <linecard#|fixedsystem>

- **ACE-6-ACTION_NOT_AVAILABLE:**
  - When a CPU traffic policy or security ACL uses an action that is not supported by the TCAM profile, it is rejected. A syslog message is generated:
    ACE-6-ACTION_NOT_AVAILABLE: The <action> action configured in the <name> policy map is not in the TCAM action set

Other messages:

Changing the TCAM profile may restart multiple agents, and the following syslog is expected:
SAND-6-EXPECTED_AGENT_EXIT

Agent Restarts

In addition, **changing the TCAM profile may restart multiple agents**, so these Syslogs are expected:

- ***SAND-6-EXPECTED_AGENT_EXIT***

Limitations

The platform does not support any arbitrarily created TCAM profile. If the TCAM profile cannot be programmed, the show command will print 'ERROR' in the Status column. Please contact Arista if the profile with the required features does not work. An user created TCAM profile after system upgrade may not support the new features introduced in the new release. For example, after upgrading from 4.21.0 to 4.22.0, TCAM profile profile1 created in 4.21.0 may not support the new features introduced in 4.22.0. If the new features are required, please recreate the profile in the new release.

One limitation is that even if the profile works, it may not be compatible with other configurations. For example, if the profile does not include Port Acl, the port acl configuration will fail. Please ensure that the profile contains all the features required by the configuration.

Future Compatibility

To ensure that customer profiles continue to work in future releases, provide the TCAM profile config to Arista engineering, so it can be validated in a test environment.

Software Downloads (/en/support/software-download)

News

News Room (/en/company/news/in-the-news)

Events (/en/company/news/events)

Blogs (/blogs)

About Arista

Company (/en/company/company-overview)

Management Team (/en/company/management-team)

Careers (/en/careers)

Investor Relations (http://investors.arista.com/)