了解Catalyst 6500 S2T CEF条目

目录

[简介](#)

- [先决条件](#)
- [要求](#)
- [使用的组件](#)

[背景信息](#)

[网络图](#)

[识别分布式转发引擎上的CEF条目](#)

- [删除CEF条目](#)

[添加CEF条目](#)

- [添加和删除VRF路由表的条目](#)

简介

本文档介绍具有管理引擎Sup2T的Cisco Catalyst 6500如何在用于实现数据包转发的线卡硬件中对Cisco IOS软件上配置的（Cisco快速转发）CEF条目进行编程。

先决条件

要求

Cisco 建议您了解以下主题：

- Cisco 快速转发 (CEF)
- Cisco Catalyst 6500 系列交换机
- 思科分布式转发卡(DFC)

使用的组件

本文档中的信息基于下列硬件和软件版本：

- Cisco Catalyst 6500 WS-X6848-GE-TX（带DFC4）线卡。
- IOS 15.2.1SY5版上带有Supervisor 2T的Cisco Catalyst 6500

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您的网络处于活动状态，请确保您了解所有命令的潜在影响。

背景信息

大多数Cisco多层交换机都使用CEF作为第3层交换机制。网络工程师必须了解CEF的工作方式，以便每天对网络故障、丢包或数据包延迟情况进行故障排除。

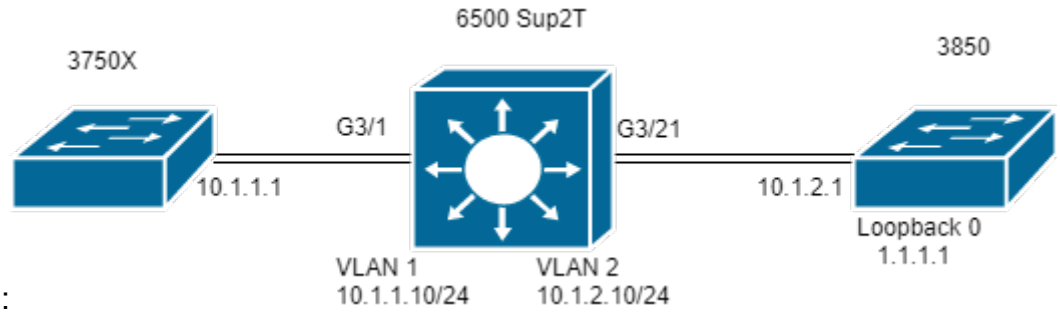
独立模式或VSS模式下的Sup2T Supervisor当前被许多企业网络部署为核心交换机，实际上汇聚了所有其他路由或交换设备。这也意味着转发大多数域内和域间流量，以便成功将数据包传送到目的地。为此，Sup2T必须具备通过静态或路由协议动态获取的正确路由信息。

在模块化机箱中，除了管理引擎之外，可能还存在多个转发引擎。为了提高数据包交换性能，某些线路卡（特别是新一代的线路卡，如C6800-32P10G）已经包含自己的转发引擎，CEF条目的查找在本地执行，导致资源以最佳方式分配到通过不同线路卡进入的流量。这些称为分布式转发卡(DFC)。

所有转发引擎共享的这些CEF条目可能由于多种原因未能在HW中分配，从软件缺陷状况、资源耗尽到高CPU状况，并阻止交换机有足够的时间更新所有条目，这可能导致一系列不希望发生的事件。

网络图

网络:




```
Switch#show module 3
---------------------- -----------------------------
Mod Ports Card Type Model Serial No.
--- ----- -------------------------------------- ------------------ -----------
3 48 CEF720 48 port 10/100/1000mb Ethernet WS-X6848-GE-TX SAL2003X5AH
---- -------------------------- ------------------ ----------- ------- -------
3 Distributed Forwarding Card WS-F6K-DFC4-A SAL2003X5AH 1.4 Ok
```

识别分布式转发引擎上的CEF条目

在图中，一台独立的6506交换机安装了一个Supervisor 2T以及一个线卡WS-6848-GE-TX，插槽3中有一个DFC。通过端口G3/1连接到线卡的主机3750X将流量发送到3850的Loopback 0地址1.1.1.1。

为此，3750X具有到IP地址1.1.1.1的静态路由，该路由通过下一跳10.1.1.10到达，即Sup2T交换机中VLAN 1的SVI。Sup2T交换机需要根据IP 1.1.1.1/32的静态路由条目通过下一跳10.1.2.1（连接到VLAN 2中Sup2T的3850接口）将此流量路由到3850交换机。

```
MXC.CALO.3750X#show ip route | inc 1.1.1.1
S 1.1.1.1 [1/0] via 10.1.1.10
```

```
MXC.CALO.Sup2T#show ip route | inc 1.1.1.1
S 1.1.1.1 [1/0] via 10.1.2.1


CALO.MXC.3850#show ip route | inc 1.1.1.1
C 1.1.1.1 is directly connected, Loopback1
```

请注意，为简单起见，3750X和3850交换机均通过同一线卡连接到6500。这意味着流量会在本地查找，并在本地转发。

数据包通过Gi3/1进入Sup2T交换机，最终到达转发引擎（因为这是一个DFC）。 转发引擎解析此数据包中的目标IP地址字段，并在编程的CEF条目上查找最佳匹配（最长掩码）。

由于这是DFC卡，这意味着它有自己的CEF条目并对其进行验证，因此我们有必要使用命令attach [dec]或attach switch [1-2] mod [dec] for VSS连接到线卡。

现在，您应该在DFC提示符下，使用命令show platform hardware cef或show platform hardware cef vpn 0返回为常规路由表编程的所有CEF条目（VPN 0/无VRF）。

由于目标是前缀1.1.1.1/32，因此您使用命令show platform hardware cef vpn 0 lookup 1.1.1.1。该命令返回前缀1.1.1.1的最佳匹配和它用于实际转发流量的最佳匹配：

<#root>

```
MXC.CALO.Sup2T#attach 3
Trying Switch ...
Entering CONSOLE for Switch
Type "^C^C^C" to end this session

MXC.CALO.Sup2T-dfc3#

show platform hardware cef vpn 0

Codes: decap - Decapsulation, + - Push Label
Index Prefix Adjacency
32 0.0.0.0/32 receive
33 255.255.255.255/32 receive
34 10.1.85.254/32 glean
35 10.1.85.5/32 receive
36 10.1.86.5/32 receive
[snip...]

MXC.CALO.Sup2T-dfc3#

show platform hardware cef vpn 0 lookup 1.1.1.1

Codes: decap - Decapsulation, + - Push Label
Index Prefix Adjacency
262 1.1.1.1/32 Vl2 ,0c11.678b.f6f7
```

CEF条目已存在，它是通过命令ip route 1.1.1.1 255.255.255.255 10.1.2.1在IOS软件中编程的静态条目而编程的。

您还可以通过命令show platform hardware cef 1.1.1.1 detail验证此条目获得命中且流量随此条目一起转发，该命令将返回邻接条目：

<#root>

MXC.CALO.Sup2T-dfc3#

show platform hardware cef 1.1.1.1 detail

```
Codes: M - mask entry, V - value entry, A - adjacency index, NR- no_route bit
LS - load sharing count, RI - router_ip bit, DF: default bit
CP - copy_to_cpu bit, AS: dest_AS_number, DGTv - dgt_valid bit
DGT: dgt/others value

Format:IPV4 (valid class vpn prefix)
M(262 ): 1 F 2FFF 255.255.255.255
V(262 ): 1 0 0 1.1.1.1
(A:114689, LS:0, NR:0, RI:0, DF:0 CP:0 DGTv:1, DGT:0)
```

最后，邻接条目显示数据包如何重写，以及流量是否被此邻接条目重写：

<#root>

MXC.CALO.Sup2T-dfc3#

show platform hardware cef adjacencies entry 114689 detail

```
RIT fields: The entry has a Layer2 Format
_____________________________________________________________
|decr_ttl = YES | pipe_ttl = 0 | utos = 0
|_____________________|_____________________|_____________________
|l2_fwd = 0 | rmac = 0 | ccc = L3_REWRITE
|_____________________|_____________________|_____________________
|rm_null_lbl = YES| rm_last_lbl = YES| pv = 0
|_____________________|_____________________|_____________________
|add_shim_hdr= NO | rec_findex = N/A | rec_shim_op = N/A
|_____________________|_____________________|_____________________
|rec_dti_type = N/A | rec_data = N/A
|_____________________________________|_____________________
|modify_smac = YES| modify_dmac = YES| egress_mcast = NO
|_____________________________________|_____________________
|ip_to_mac = NO
|_____________________________________________________________
|dest_mac = 0c11.678b.f6f7 | src_mac = d8b1.902c.9680
|_____________________________|_____________________________
|
Statistics: Packets = 642
Bytes = 75756                    <<<<
```

dest_mac和src_mac是主要关注值，表示为此数据包写入的新L2报头。目的MAC地址0c11.678b.f6f7为10.1.2.1，即3850（到达1.1.1.1的下一跳）：

```
MXC.CALO.Sup2T#show ip arp 10.1.2.1
Protocol Address Age (min) Hardware Addr Type Interface
Internet 10.1.2.1 30 0c11.678b.f6f7 ARPA Vlan2
```

此外，Statistics字段显示流量实际到达此邻接条目并相应地重写L2报头。

删除CEF条目

删除CEF条目可以帮助我们删除任何可能编程错误（例如，指向错误的邻接条目）或甚至用于培训目的的条目。它还提供了修改路由路径的方法。

要删除CEF条目，您需要了解CEF条目按顺序编程，并分配了硬件索引，例如：

MXC.CALO.Sup2T-dfc3#show platform hardware cef vpn 0

代码：decap — 解封，+ — 推送标签

<#root>

```
MXC.CALO.Sup2T-dfc3#show platform hardware cef vpn 0
...
Index Prefix Adjacency
259 10.1.2.255/32 receive
260 10.1.1.1/32 Vl1 ,a0ec.f930.3f40
261 10.1.2.1/32 Vl2 ,0c11.678b.f6f7
262
 1.1.1.1/32 Vl2 ,0c11.678b.f6f7
    <<<< Our CEF entry of interest has a HW index of 262.
...
```

此硬件索引是删除CEF条目的最重要元素，因为它用作参考。但是，要对它进行任何更改，必须将其转换为软件句柄。您可以使用命令test platform hardware cef index-conv hw_to_sw [hw index]实现此目的

<#root>

```
MXC.CALO.Sup2T-dfc3#
test platform hardware cef index-conv hw_to_sw 262
hw index: 262
---->
 sw handle: 101
```

现在您已经知道了软件句柄，您可以继续使用test platform hardware cef v4-delete [sw handle] mask [mask length] vpn [dec]命令删除CEF条目

```
<#root>

MXC.CALO.s2TVSS-sw2-dfc3#

test platform hardware cef v4-delete 101 mask 32 vpn 0

test_ipv4_delete: done.
```

注意：掩码长度值为32，因为这是特定于主机的路由(1.1.1.1/32)

现在，我们的CEF条目被删除：

```
<#root>

MXC.CALO.Sup2T-dfc3#

show platform hard cef vpn 0 1.1.1.1

Codes: decap - Decapsulation, + - Push Label
Index Prefix Adjacency


MXC.CALO.Sup2T-dfc3#

show platform hard cef vpn 0

[snip...]
259 10.1.2.255/32 receive
260 10.1.1.1/32 Vl1 ,a0ec.f930.3f40

261 10.1.2.1/32 Vl2 ,0c11.678b.f6f7
288 224.0.0.0/24 receive              <<<<<<< Index 262 no longer exists in the CEF entries.

289 10.1.85.0/24 glean
```

请注意，test platform hardware cef vpn 0命令是在DFC提示符下执行的。这样，CEF条目已从DFC的CEF表中删除，而不是从Supervisor中删除，因此，您必须非常小心从哪个转发引擎中删除这些条目。

流量发生变化时存在不可见性风险（在实验室测试中），这可能是由于另一个CEF条目被命中所致。考虑始终匹配最精确的一个（最长掩码）。 在本实验中，它命中：

```
<#root>

MXC.CALO.Sup2T-dfc3#

show plat hard cef vpn 0 lookup 1.1.1.1

Codes: decap - Decapsulation, + - Push Label
Index Prefix Adjacency
262048 0.0.0.0/0 glean
```

那么，此条目实际上对数据包有何作用？:

```
MXC.CALO.Sup2T-dfc3#show platform hardware cef adjacencies entry 262048
RIT fields: The entry has a Recirc. Format
_____________________________________________________________
|decr_ttl=NO | l2_fwd=NO | ccc = 6 | add_shim_hdr = YES
|____________|___________|_________|_____________________
|rc_fidx=0 | rc_shimop=1 | rc_dti_type=4 | rc_data = 0x10B
|____________|_____________|_________________|_____________

Statistics: Packets = 2163
Bytes = 255234
```

<#root>

Taken from a CPU packet capture using Catlayst 6500 NETDR tool. For NETDR capture tool details refer to

------- dump of incoming inband packet -------

```
l2idb Po1, l3idb Vl1, routine inband_process_rx_packet, timestamp 01:00:17.841
dbus info: src_vlan 0x1(1), src_indx 0xB40(2880), len 0x82(130)
bpdu 0, index_dir 0, flood 0, dont_lrn 0, dest_indx 0x5FA4(24484), CoS 0
cap1 0, cap2 0
78020800 00018400 0B400100 82000000 1E000464 2E000004 00000010 5FA45BDD
destmac D8.B1.90.2C.96.80, srcmac A0.EC.F9.30.3F.40, shim ethertype CCF0
earl 8 shim header IS present:
version 0, control 64(0x40), lif 1(0x1), mark_enable 1,
feature_index 0, group_id 0(0x0), acos 0(0x0),
ttl 14, dti 4, dti_value 267(0x10B)
10000028 00038080 010B
ethertype 0800
protocol ip: version 0x04, hlen 0x05, tos 0x00, totlen 100, identifier 51573
df 0, mf 0, fo 0, ttl 255, src 10.1.1.1, dst 1.1.1.1
icmp type 8, code 0
```

------- dump of outgoing inband packet -------

```
l2idb NULL, l3idb Vl2, routine etsec_tx_pak, timestamp 01:03:56.989
dbus info: src_vlan 0x2(2), src_indx 0x380(896), len 0x82(130)
bpdu 0, index_dir 0, flood 0, dont_lrn 0, dest_indx 0x0(0), CoS 0
cap1 0, cap2 0
00020000 0002A800 03800000 82000000 00000000 00000000 00000000 00000000
destmac 0C.11.67.8B.F6.F7, srcmac D8.B1.90.2C.96.80, shim ethertype CCF0
earl 8 shim header IS present:
version 0, control 0(0x0), lif 16391(0x4007), mark_enable 0,
feature_index 0, group_id 0(0x0), acos 0(0x0),
ttl 15, dti 0, dti_value 540674(0x84002)
000800E0 0003C008 4002
ethertype 0800
protocol ip: version 0x04, hlen 0x05, tos 0x00, totlen 100, identifier 50407
df 0, mf 0, fo 0, ttl 254, src 10.1.1.1, dst 1.1.1.1
icmp type 8, code 0
```

现在，所有通过线路卡3进入的目的地为1.1.1.1的流量都会通过填充码报头重新循环并传送到CPU。有时，不是此CEF条目，而是看到另一个具有drop邻接的0.0.0.0/0，并且执行完全相同的操作。

注意：评估删除的CEF条目。CPU使用率过高可能是由于此原因。通常配置默认路由0.0.0.0/0，并根据该路由转发流量（并导致数据包丢失）。

添加CEF条目

添加CEF条目时，大多数情况下会解决导致数据包丢失、数据包延迟或CPU使用率较高的任何编程错误问题。了解如何在硬件中安装CEF条目，不仅能够纠正编程错误的条目，而且能够通过数据包的重新循环操作任何数据包转发，将其指向完全不同的接口或下一跳，根据需要重写路由的数据包和/或丢弃数据包等。所有这些，无需重新加载机箱，即可删除并设置配置或任何明显的更改。无需进入配置模式即可添加CEF条目。（正如上一节中介绍的CEF条目删除步骤一样）。

基本上，这里有两种情况：您有到下一跳的有效ARP条目（在本例中为10.1.2.1），以及您没有（出于任何原因）。 第二种情况是强制您实际创建有效的ARP条目（通过静态ARP）：

步骤1.交换机中有一个10.1.2.1的ARP条目，该条目是1.1.1.1的下一跳。

<#root>

```
MXC.CALO.Sup2T#

show ip arp 10.1.2.1

Protocol Address Age (min) Hardware Addr Type Interface
Internet 10.1.2.1 2 0c11.678b.f6f7 ARPA Vlan2

MXC.CALO.Sup2T#show ip route | inc 1.1.1.1
S 1.1.1.1 [1/0] via 10.1.2.1
```

ARP条目在CEF表中被编程为主机路由(/32):

<#root>

```
MXC.CALO.Sup2T-dfc3#

show plat hard cef vpn 0 look 10.1.2.1

Codes: decap - Decapsulation, + - Push Label
Index Prefix Adjacency
53 10.1.2.1/32 Vl2 ,0c11.678b.f6f7

And of course, there is an index for this which again will tell us how a packet should be rewritten to r


MXC.CALO.Sup2T-sw2-dfc3#

show plat hard cef vpn 0 10.1.2.1 detail
[snip...]

Format:IPV4 (valid class vpn prefix)
M(53 ): 1 F 2FFF 255.255.255.255
```

```
V(53 ): 1 0 0 10.1.2.1
(A:114689, LS:0, NR:0, RI:0, DF:0 CP:0 DGTv:1, DGT:0)
```

Wait, wasn't 114689 adj entry the same used for 1.1.1.1?:

```
MXC.CALO.Sup2T-sw2-dfc3#show plat hard cef 1.1.1.1 de
[snip...]
Format:IPV4 (valid class vpn prefix)
M(54 ): 1 F 2FFF 255.255.255.255
V(54 ): 1 0 0 1.1.1.1
(A:114689, LS:0, NR:0, RI:0, DF:0 CP:0 DGTv:1, DGT:0)
```

具有任何目的IP地址且数据链路下一跳相同的任何数据包应通过同一接口转发，并使用相同的L2报头重写。

尽管这乍看上去似乎很明显，但实际上它是添加CEF条目的最重要元素，但您需要告知它如何使用特定CEF邻接条目来重写数据包。

步骤2.现在，假设没有为此自动创建ARP条目，因此您需要创建静态ARP条目。

为此，您需要知道用作前缀10.1.2.1的下一跳的设备的MAC地址，以便将其发送到0c11.678b.f6f7。如果show mac address-table address 0c11.678b.f6f7命令输出中已有一个MAC地址条目正常，则您需要创建静态MAC条目：

<#root>

```
MXC.CALO.Sup2T(config)#
```

mac address-table static 0c11.678b.f6f7 vlan 2 int Gi3/21

```
Displaying entries from DFC switch [2] linecard [3]:

vlan mac address type learn age ports
----+----+--------------+-------+-----+----------+----------------------------
2 0c11.678b.f6f7 static No - Gi3/21
```

步骤3.最后，需要创建静态ARP条目才能对CEF条目进行编程：

<#root>

```
MXC.CALO.Sup2T(config)#
```

arp 10.1.2.1 0c11.678b.f6f7 arpa

```
    <<< Static ARP configuration

MXC.CALO.Sup2T#
```

show ip arp 10.1.2.1

```
Protocol Address Age (min) Hardware Addr Type Interface
Internet 10.1.2.1 - 0c11.678b.f6f7 ARPA                  <<< Now the static ARP entry is complete

// Attaching to DFC3...
```

```
MXC.CALO.Sup2T-sw2-dfc3#
```

show plat hard cef 10.1.2.1 detail

```
[snip...]
Format:IPV4 (valid class vpn prefix)
M(53 ): 1 F 2FFF 255.255.255.255
V(53 ): 1 0 0 10.1.2.1
(A:114689, LS:0, NR:0, RI:0, DF:0 CP:0 DGTv:1, DGT:0)

The ARP entry exist in CEF table for DFC3. Same Adjacency Index result as before...
```

现在您已了解这些邻接条目的作用，您终于可以继续添加CEF条目了。在最后一部分，前缀1.1.1.1/32的CEF条目通过test platform hardware cef v4-delete命令删除。现在，通过命令test platform hardware cef v4-insert [prefix] [mask length] vpn [vpn number] adjacency [adjacency index]将其添加回

要验证这一点，请使用命令test platform hardware cef v4-insert 1.1.1.1 32 vpn 0 adjacency 114689。该条目已添加回DFC CEF表中：

<#root>

```
MXC.CALO.Sup2T-sw2-dfc3#
```

test platform hardware cef v4-insert 1.1.1.1 32 vpn 0 adjacency 114689

test_ipv4_insert: done: sw_index = 42

```
MXC.CALO.Sup2T-sw2-dfc3#
```

show plat hard cef vpn 0 1.1.1.1

```
Codes: decap - Decapsulation, + - Push Label
Index Prefix Adjacency
54 1.1.1.1/32 Vl2 ,0c11.678b.f6f7

Ping from the 3750X to Loopback 0 is successful and HW forwarded by 6500 DFC.

MXC.CALO.Sup2T-sw2-dfc3#show platform hard cef adj entry 114689

Index: 114689 -- Valid entry (valid = 1) --

RIT fields: The entry has a Layer2 Format
_____________________________________________________
|decr_ttl=YES | l2_fwd=NO | ccc = 4 | add_shim_hdr = NO
|_____________|_____________|_________|_____________________

Statistics: Packets = 684
Bytes = 80712
```

// Logs in 3850

```
CALO.MXC.385024XU#show logging
[snip...]
*Jan 23 05:59:56.911: ICMP: echo reply sent, src 1.1.1.1, dst 10.1.1.1, topology BASE, dscp 0 topoid 0
*Jan 23 05:59:57.378: ICMP: echo reply sent, src 1.1.1.1, dst 10.1.1.1, topology BASE, dscp 0 topoid 0
*Jan 23 05:59:57.390: ICMP: echo reply sent, src 1.1.1.1, dst 10.1.1.1, topology BASE, dscp 0 topoid 0
```

添加和删除VRF路由表的条目

通过前面所有步骤进行的配置，show platform hardware cef命令中的vpn 0字符串已被强制执行。尽管由于该命令默认情况下会返回常规路由表或vpn 0的条目，因此看起来完全不必要，但这样做是为了始终牢记通过您添加和删除CEF条目1.1.1.1/32的文档从特定路由表实例(VRF)添加或删除条目。但是，某些前缀很可能存在于不同的VRF(即e.10.x.x.x)以及删除、添加或修改错误的VRF的CEF条目会导致负面影响。

为VRF TEST_VRF删除前缀为1.1.1.1/32的CEF条目。有关添加CEF条目的详细说明，请参阅本文档的添加CEF条目部分。

要添加VRF，请使用命令ip vrf forwarding [VRF-NAME]将6500交换机中的SVI更改为建议的VRF，最后在TEST_VRF表中添加相同的静态路由：

<#root>

```
MXC.CALO.Sup2T(config)#ip vrf TEST_VRF
MXC.CALO.Sup2T(config-vrf)#int vlan 1
MXC.CALO.Sup2T(config-if)#ip vrf forwarding TEST_VRF
% Interface Vlan1 IPv4 disabled and address(es) removed due to enabling VRF TEST_VRF
MXC.CALO.Sup2T(config-if)#ip add 10.1.1.10 255.255.255.0
MXC.CALO.Sup2T(config-if)#int vlan 2
MXC.CALO.Sup2T(config-if)#ip vrf forwarding TEST_VRF
% Interface Vlan2 IPv4 disabled and address(es) removed due to enabling VRF TEST_VRF
MXC.CALO.Sup2T(config-if)#ip add 10.1.2.10 255.255.255.0
MXC.CALO.Sup2T(config)#ip route vrf TEST_VRF 1.1.1.1 255.255.255.255 10.1.2.1

MXC.CALO.Sup2T#
```

show ip vrf

```
 Name Default RD Interfaces

TEST_VRF

 <not set> Vl1
 Vl2
```

VRF也按顺序编程。这是交换机中的第一个VRF（之前未配置其他VRF），因此此VRF实例的vpn编号应为1。请运行show platform hardware cef vpn 1命令以验证是否正确：

<#root>

```
MXC.CALO.Sup2T-sw2-dfc3#

show plat hard cef vpn 1

Codes: decap - Decapsulation, + - Push Label
Index Prefix Adjacency
34 10.1.1.10/32 receive
```

```
35 10.1.1.0/32 receive
36 10.1.1.255/32 receive
38 10.1.2.10/32 receive
43 10.1.2.0/32 receive
44 10.1.2.255/32 receive
53 10.1.2.1/32 Vl2 ,0c11.678b.f6f7
```

54 1.1.1.1/32 Vl2 ,0c11.678b.f6f7

```
[snip...]
```

**However, usually, switches have hundred or thousands of VRFs and just count them in the 'show ip vrf' co
it will return the actual vpn number for that VRF:**

```
Format:IPV4 (valid class vpn prefix)
M(54 ): 1  F 2FFF 255.255.255.255
V(54 ): 1  0
```

1

```
    1.1.1.1                                <<<<<<<<<<< The number in red determines the VPN this prefix
(A:114689, LS:0, NR:0, RI:0, DF:0 CP:0 DGTv:1, DGT:0)
```

了解此条目的实际VPN编号和软件索引非常重要，以便您可以继续将其从此VRF实例中删除或添加到此VRF实例：

<#root>

```
MXC.CALO.Sup2T-sw2-dfc3#
```

test platform hardware cef index-conv hw_to_sw 54

```
hw index: 54 ----> sw handle: 42

MXC.CALO.Sup2T-sw2-dfc3#
```

test platform hardware cef v4-delete 42 mask 32 vpn 1

test_ipv4_delete: done.

```
Result:

MXC.CALO.Sup2T-sw2-dfc3#
```

show platform hardware cef vpn 1 lookup 1.1.1.1

```
Codes: decap - Decapsulation, + - Push Label
Index Prefix Adjacency
```

262049 0.0.0.0/0 drop

Traffic is now getting punted, and the effects are seen in the 3750X pings to 1.1.1.1:

MXC.CALO.3750X#ping 1.1.1.1 repe 5000000

```
Sending 5000000, 100-byte ICMP Echos to 1.1.1.1, timeout is 2 seconds:
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!.!!!!!!!!!!!!!!!!
[snip...]

// Packet loss
```

假设在生产网络中，由于这些CEF条目条件，出现了丢包和音频或视频不连续的情况。因此，建议在维护窗口中执行这些测试。

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。