

Deploy ELAM to Capture VXLAN Encapsulation Packets on Nexus 7000 Series Switches

Contents

[Introduction](#)

[Background Information](#)

[Topology](#)

[Configure the Trigger](#)

[Interpret the Results](#)

[Related Information](#)

Introduction

This document describes how to deploy the Embedded Logic Analyzer Module (ELAM) to capture VXLAN Encapsulation Packets on Nexus 7000 Series Switches.

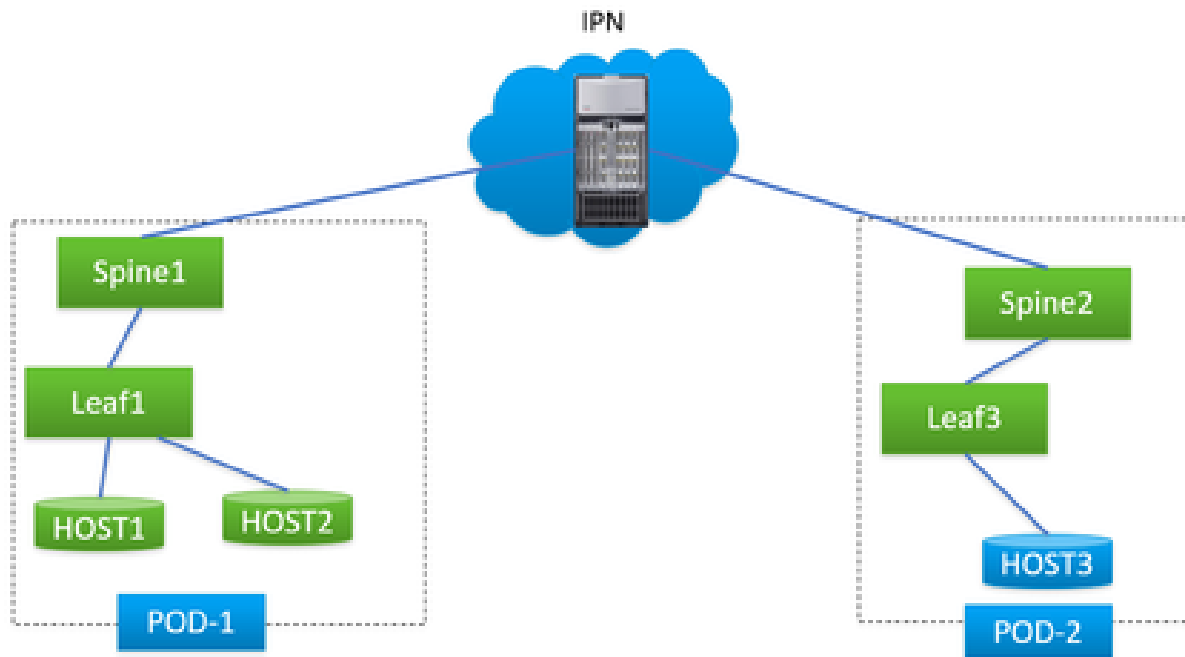


Tip: Refer to the [ELAM Overview](#) document for an overview of ELAM.

Background Information

Many users are currently utilizing N7K as an IPN/ISN transit device for their ACI MPOD/MSITE deployment. However, when compared to N9K, N7K lacks the robust capability to set the ELAM trigger based on a rich Outer(12(vntag)|13|14)-inner(12|13|14)-ieth combination. As a result, it becomes challenging to determine if a specific VXLAN-encapsulated packet is hitting the N7K at the IPN edge from an ELAM perspective. This document outlines a method to address this challenge.

Topology



In this scenario, a straightforward ACI MPOD topology is illustrated, where IPN is an N7K with an F3 card. HOST1 and HOST2 are in pod1, HOST3 is in pod2. HOST1 can communicate with HOST3, but HOST2 cannot. Following troubleshooting conducted by an ACI engineer, it was determined that the packets from HOST2 to HOST3 were sent out to N7K from spine1 in pod1 but were never received by spine2 in pod2. This was verified through ELAM on ACI spines, leading to the suspicion that the packets were being dropped on N7K.

Is it possible to definitively attribute the issue to the N7K based solely on the ELAM results on ACI spines? Certainly not. The ELAM on egress spine1 indicated that it sent the packet to N7K, but this does not guarantee that the packet physically reached N7K, as packets may still be dropped after the ELAM cycle due to lower-layer issues. However, when you ELAM these specific packets on the N7K side, it can assist us in accurately identifying the correct device involved in the issue.

Configure the Trigger

'13-packet-length' is a valid ELAM trigger for nearly all different generation LCs on the N7K platform. Therefore, let's utilize it to establish the ELAM condition on the IPN N7K. The task involves controlling the HOST to transmit testing packets with a specified packet length, as illustrated:

```
<#root>
```

```
#
```

```
ping 172.28.1.20 packet-size 777
```

```
PING 172.28.1.20 (172.28.1.20): 777 data bytes
```

```
785 bytes from 172.28.1.20: icmp_seq=0 ttl=252 time=1.246 ms
```

```
785 bytes from 172.28.1.20: icmp_seq=1 ttl=252 time=0.846 ms
```

```
785 bytes from 172.28.1.20: icmp_seq=2 ttl=252 time=0.84 ms
```

```
785 bytes from 172.28.1.20: icmp_seq=3 ttl=252 time=0.814 ms
```

```
785 bytes from 172.28.1.20: icmp_seq=4 ttl=252 time=0.817 ms
```

```
--- 172.28.1.20 ping statistics ---
```

```
5 packets transmitted, 5 packets received, 0.00% packet loss
```

```
round-trip min/avg/max = 0.814/0.912/1.246 ms
```

The PING utility is integrated into any type of OS, with only slight variations in parameters based on the OS you are using. One crucial point to highlight is to pay attention to the packet size you specify when initiating the PING in your OS. In this example, the 777B represents the pure data length, requiring an additional 8B (ICMP header) and 20B (IP header) to obtain the final IP length of 805B. After VXLAN encapsulation (adding an extra 50B overhead), you can anticipate the packet hitting the N7K at 855B. Let's configure it in ELAM.

In this example, the interface that connects to spine1 is E7/1 and E7/4 connects to spine2.

```
<#root>
```

```
#
```

```
show module 7
```

Mod	Ports	Module-Type	Model	Status
7	12	10/40 Gbps Ethernet Module	N7K-F312FQ-25	ok

```
module-7#
```

```
show hardware internal dev-port-map
```

```
-----  
CARD_TYPE:      12 port 40G
```

```
>Front Panel ports:12
```

```
-----  
Device name      Dev role      Abbr num_inst:  
-----
```

```
> Flanker Eth Mac Driver DEV_ETHERNET_MAC      MAC_0 6
```

```

> Flanker Fwd Driver      DEV_LAYER_2_LOOKUP    L2LKP  6
> Flanker Xbar Driver     DEV_XBAR_INTF         XBAR_INTF 6
> Flanker Queue Driver   DEV_QUEUEING          QUEUE   6
> Sacramento Xbar ASIC   DEV_SWITCH_FABRIC     SWICHF  1
> Flanker L3 Driver       DEV_LAYER_3_LOOKUP    L3LKP  6
> EDC                     DEV_PHY               PHYS    2

```

```

+-----+
+-----+++FRONT PANEL PORT TO ASIC INSTANCE MAP+++-----+
+-----+

```

```
FP port |  PHYS |  MAC_0 |  L2LKP |  L3LKP |  QUEUE |SWICHF
```

1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	1	1	1	1	1	0
4	1	1	1	1	1	0
5	0	2	2	2	2	0
6	0	2	2	2	2	0
7	1	3	3	3	3	0
8	1	3	3	3	3	0
9	4	4	4	4	4	0
10	4	4	4	4	4	0
11	5	5	5	5	5	0
12	5	5	5	5	5	0

```

+-----+
+-----+

```

So, you need to set it up in instance 0.

```
<#root>
```

```
module-7# elam asic flanker instance 0
```

```
module-7(fln-elam)# layer2
module-7(fln-l2-elam)#
trigger dbus ipv4 ingress if l3-packet-length 855

module-7(fln-l2-elam)#
trigger rbus ingress if trig

module-7(fln-l2-elam)# start
module-7(fln-l2-elam)# status
ELAM Slot 7 instance 0: L2 DBUS Configuration: trigger dbus ipv4 ingress if l3-packet-length 855
L2 DBUS: Triggered
ELAM Slot 7 instance 0: L2 RBUS Configuration: trigger rbus ingress if trig
L2 RBUS: Triggered
```

Interpret the Results

```
<#root>
```

```
module-7(fln-l2-elam)#
```

```
show dbus
```

```
cp = 0x10084d00, buf = 0x10084d00, end = 0x10091050
```

```
-----
Flanker Instance 00 - Capture Buffer On L2 DBUS:
```

```
Status(0x1102), TriggerWord(0x000), SampleStored(0x008),CaptureBufferPointer(0x000)
```

```
is_l2_egress: 0x0000, data_size: 0x023
```

```
[000]: 14f4a000 08010000 00000000 6d200800 00006000 00000000 01800100 00000000 00000000 00000000 00003000
```

```
0590 00990000 00000000 00000000 00000005 88405000 00000000 00000000 00000000 00000000 00000000 00000000
```

```
a4 2dbeef00
```

```
Printing packet 0
```

L2 DBUS PRS MLH IPV4

label-count	: 0x0	mc	: 0x0
null-label-valid	: 0x0	null-label-exp	: 0x0
null-label-ttl	: 0x0	1b10-vld	: 0x0
1b10-eos	: 0x0	1b10-1b1	: 0x0
1b10-exp	: 0x0	1b10-ttl	: 0x0
1b11-exp	: 0x0	1b11-ttl	: 0x0
ipv4	: 0x0	ipv6	: 0x0
14-protocol	: 0x11		
df	: 0x0		
mf	: 0x0	frag	: 0x0
t11	: 0x1f	13-packet-length	: 0x357
option	: 0x0	tos	: 0x0
sup-eid	: 0x0	header-type	: 0x1
error	: 0x0	redirect	: 0x0
port-id	: 0x0	last-ethertype	: 0x800
12-frame-type	: 0x0	da-type	: 0x0
packet-type	: 0x0	12-length-check	: 0x0
ip-da-multicast	: 0x0	ip-multicast	: 0x0
ip-multicast-control	: 0x0	ids-check-fail	: 0x0
tr	: 0x0	outer-cos	: 0x0
inner-cos	: 0x0	vqi-valid	: 0x0
vqi	: 0x0	packet-length	: 0x369
vlan	: 0x4	destination-index	: 0x0
source-index	: 0x30		
bundle-port	: 0x0		
acos	: 0x0	outer-drop-eligibility	: 0x0

inner-drop-eligibility: 0x0	sg-tag	: 0x0	
rbh	: 0x0	vs1-num	: 0x0
inband-flow-creation-deletion: 0x0	ignore-qoso	: 0x0	
ignore-qosi	: 0x0	ignore-aclo	: 0x0
ignore-acli	: 0x0	index-direct	: 0x0
no-stats	: 0x0	dont-forward	: 0x0
notify-index-learn	: 0x1	notify-new-learn	: 0x1
disable-new-learn	: 0x0	disable-index-learn	: 0x0
dont-learn	: 0x0	bpdu	: 0x0
ff	: 0x0	rf	: 0x0
ccc	: 0x0	l2	: 0x0
rdt	: 0x0	dft	: 0x0
dfst	: 0x0	status-ce-1q	: 0x0
status-is-1q	: 0x1	trill-encap	: 0x0
mim-valid	: 0x0	dtag-ttl	: 0x0
dtag-ftag	: 0x0	valid	: 0x1
erspan-kpa-valid	: 0x0	recir-shim-vxlan-src-peer-id: 0x0	
vn-valid	: 0x0	source-vif	: 0x0
destination-vif	: 0x0	vn-p	: 0x0
sequence-number	: 0x60	v1	: 0x0
inner-de-valid	: 0x0	de-cfi	: 0x0
second-inner-cos	: 0x0	tunnel-type	: 0x2

UDP OTV/LISP TUNNEL BNDL

vlan-tag-valid: 0x0	segment-id-valid: 0x0
v1: 0x0	de: 0x0
sgt-valid: 0x0	inner-ip-ttl: 0x0
ip-da-multicast: 0x0	
lisp-inst-id: 0x2c8004	
lisp-flags: 0xc8	isis-mac-da-valid: 0x0

```
type: 0x0
shim-valid      : 0x0
segment-id-valid : 0x0          copp          : 0x0
dti-type-vpnid  : 0x0          segment-id   : 0x0
ib-length-bundle : 0x58840     m1h-type     : 0x5
ulh-type        : 0x4
```

```
source-ipv4-address: 10.0.200.64
```

```
destination-ipv4-address: 10.1.224.67
```

```
mim-destination-mac-address : 0000.0000.0000
```

```
mim-source-mac-address : 0000.0000.0000
```

```
destination-mac-address : 00c1.b1c9.c2c4
```

```
source-mac-address : 000d.0d0d.0d0d
```

Because l3 packet length is used as the trigger, there is a possibility that the ELAM could be triggered by background packets unintended for capture. Therefore, it is imperative to utilize other fields in the capture for a double cross-check of the capture results. This ensures that the captured packet aligns with our intended criteria, including fields like source IP (sip), destination IP (dip), time-to-live (ttl), source index, etc. An interesting observation is that, although N7K doesn't support using VXLAN VNID as a trigger, in the output interpreter, the field 'lisp-inst-id: 0x2c8004' corresponds to the VNID in the VXLAN header.

```
<#root>
```

```
module-7(f1n-12-elam)# dec
```

```
0x2c8004
```

```
2916356
```

```
Leaf3#
```

```
show system internal epm endpoint ip 172.28.1.20
```

```
MAC : 0000.2222.1202 ::: Num IPs : 1
```


IP# 0 : 172.28.1.20 ::: IP# 0 flags : host-tracked| ::: l3-sw-hit: Yes ::: flags2 :

Vlan id : 186 ::: Vlan vnid : 11494 ::: VRF name : zixu:vrf

BD vnid : 16482209 :::

VRF vnid : 2916356

/* Confirming the VNID from ACI LEAF side */

Phy If : 0x1a00b000 ::: Tunnel If : 0

Interface : Ethernet1/12

Flags : 0x80005c04 ::: sclass : 16388 ::: Ref count : 5

EP Create Timestamp : 01/22/2021 15:42:49.243582

EP Update Timestamp : 02/08/2021 11:26:52.882308

EP Flags : local|IP|MAC|host-tracked|sclass|timer|

module-7(fln-12-elam)#

show rbus

cp = 0x100a96fc, buf = 0x100a96fc, end = 0x100b5a4c

Flanker Instance 00 - Capture Buffer On L2 RBUS:

Status(0x1102), TriggerWord(0x000), SampleStored(0x008),CaptureBufferPointer(0x000)

is_l2_egress: 0x0000, data_size: 0x018

[000]: 0015cb30 0000006d 20000000 03000000 00000000 00000000 00000014 2d8000a0 3c3c0000 00000000 020000

0000 00000400 00008000 005d0000 001e0002 2bd7c0cf f96002a0 000000ba

Printing packet 0

L2 RBUS INGRESS CONTENT

pad : 0x572c valid : 0x1
l2-rbus-trigger : 0x1 sequence-number : 0x60
rit-ipv4-id : 0x0 ipv4-tunnel-encap : 0x0

rit-mp1s-rw	: 0x0	m12-ptr	: 0x0
m13-ptr	: 0x0	mark	: 0x0
result-cap3	: 0x0	di1-v5-delta-length	: 0x0
di1-v5-delta-length-plus	: 0x0	di1-v4-delta-length	: 0x0
di1-v4-delta-length-plus	: 0x0	di2-delta-length	: 0x0
di2-delta-length-plus	: 0x0	m12-delta-length	: 0x0
m12-delta-length-plus	: 0x0	m13-delta-length	: 0x0
m13-delta-length-plus	: 0x0	s-vector	: 0x0
1cpu-ff-valid	: 0x0	sup-di-vqi	: 0x0
erspan-term-index-dir	: 0x0	erspan-buffer-check	: 0x0
12-tunnel-decapped	: 0x0	13-delta-length	: 0x0
rit-crc16-valid	: 0x1	rit-crc16	: 0x42d8
vntag-p	: 0x0	frr-recirc	: 0x0
ingress-lif	: 0x5	ear1-proxy-vld	: 0x0
md-di-vld	: 0x0	rc	: 0x0
segment-id-valid	: 0x0	t11-out	: 0x1e
t11-mid	: 0x1e	tos-out	: 0x0
tos-in	: 0x0	orig-vlan1	: 0x0
vlan1	: 0x0	source-peer-id	: 0x0
final-ignore-qoso	: 0x0	port-id	: 0x0
cr-type	: 0x1	pup-packet	: 0x0
bpdu	: 0x0	vdc	: 0x0
tr	: 0x0	de	: 0x0
cos	: 0x0	inner-drop-eligibility	: 0x0
inner-cos	: 0x0	acos	: 0x0
di-1t1-index	: 0x3c		
13-multicast-di	: 0x3c		
source-index	: 0x30	vlan	: 0x4
index-direct	: 0x0	di1-valid	: 0x1
vqi	: 0x4a	di2-valid	: 0x0
v5-fpoe-idx	: 0x0	di2-fpoe-idx	: 0x0

```

13-multicast-v5      : 0x0          dft                : 0x0
dfst                 : 0x0          13-learning-ff     : 0x0
result-rbh           : 0x40          di2-cr-type         : 0x0
result-2             : 0x1          dtag-ftag           : 0x0
dtag-ttl             : 0x20          mac-in-mac-op       : 0x0
dvif                 : 0x0          result-cap1         : 0x0
result-cap2          : 0x0          erspan-term         : 0x0
erspan-decap         : 0x0          dont-learn          : 0x0
routed-frame         : 0x1          copy-cause          : 0x0
12-copy-cause        : 0x0          13-rit-ptr         : 0x5d
sg-tag               : 0x0          trill-nh-id         : 0x0
ttl-in               : 0x1e          fc-up               : 0x0
up-did               : 0x0          did                 : 0x22bd
up-sid               : 0x0          sid                 : 0xf819ff
shim-12-tunnel-encap: 0x0          shim-ls-hash        : 0xb
shim-rc              : 0x0          shim-lif            : 0x5
shim-replication-pkt: 0x0          shim-router-mac     : 0x1
shim-mark-enable     : 0x0          shim-qos-group-id   : 0x0
shim-destination-table-index: 0x5d      shim-acos-preserve  : 0x0
mim-destination-mac-address : 0000.0000.0000
mim-source-mac-address  : 0000.0000.0000

```

```
module-7(f1n-12-elam)#
```

```
show system internal pixmc info 1t1-cb 1t1 0x30
```

```

1t1 | 1t1_type | if_index | 1c_type | vdc | v4_fpoe | v5_fpoe | base_fpoe_idx | flag
0x0030 | 5 | |
Eth7/1
| 2 | 4 | 0x00 | 0x00 | 0x0000 | 0x0

```

```
module-7(fln-12-elam)#
```

```
show system internal pixmc info lt1-cb lt1 0x3c
```

```
lt1 | lt1_type | if_index | lc_type | vdc | v4_fpoe | v5_fpoe | base_fpoe_idx | flag  
0x003c | 5 | |  
Eth7/4  
| 2 | 4 | 0x00 | 0x00 | 0x0000 | 0x0
```

The same methodology also works if you try to capture Broadcast, Unknown Unicast, and Multicast (BUM) packets within an ACI flooding BD, you just need to set a static ARP and point to a nonexistent MAC on your HOST, then launch the PING in the same way.

Related Information

You can refer to these links for more details on how to use ELAM on different N7K LCs:

- [ELAM Overview](#)
- [N7K M-Series Module ELAM Procedure](#)
- [N7K F1 Module ELAM Procedure](#)
- [N7K F2 Module ELAM Procedure](#)
- [N7K M3 Module ELAM Procedure](#)
- [Cisco Technical Support & Downloads](#)