Verify Layer 2 LISP Connectivity in SDA on Catalyst 9000 Switches

Contents

Introduction
Requirements
Components Used
Background Information
Layer-2 in the Fabric
Network Diagram
Verify
L2-LISP Instance Scale
Control-Plane
Step 1. Proper SVI Provisioning
Step 2. MAC address Table
Step 3. ARP Table
Step 4. LISP Database for Local Hosts
Step 5. RLOC of Remote Gosts with LISP Internet Groper (LIG)
Step 6. LISP Map-Cache for Remote Hosts
Step 7. Switch Integrated Security Features (SISF) Database (Device-Tracking Database)
Data-Plane (Encapsulation Path)
Step 1. Entries on MAC Address Table Manager (MATM)
Step 2. Print Resource Handle Information for 'machandle'
Step 3. Key VLAN (MVID)
Step 4. Group Port Number (GPN)
Step 5. Station Index (SI)
Step 6. Destination Index (DI)
Step 7. Rewrite Index (RI)
Step 8. Forwarding Decision for the New Encapsulated Packet
Data-Plane (Decapsulation Path)
Step 1. Decap Information from IFM
Step 2. Features Programmed on Tunnel Interface via L3IF LE Handle
Step 3. Station Index (SI) Handle for Tunnel Interface (Decap Process)
Step 4. Destination Index (DI) and Rewrite Index (RI) Handle for Tunnel interface (Decap Process)
Troubleshoot
Packet Captures withEmbedded Packet Capture (EPC) Tool
Related Information

Introduction

This document describes how to verify Layer-2 LISP in Software-Defined Access (SDA) on Catalyst 9000 switches.

Requirements

Cisco recommends that you have knowledge of these topics:

- Knowledge of SDA solution and the components.
- Understanding of Locator/ID Separation Protocol (LISP) - Control-Plane - and Virtual Extensible

LAN (VxLAN) - Data-Plane - protocols.
- Acquainted with Catalyst 9000 architecture.

Components Used

The information in this document is based on these software and hardware versions:

- Catalyst 9300
- Cisco IOS® XE 16.9.8 and later

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

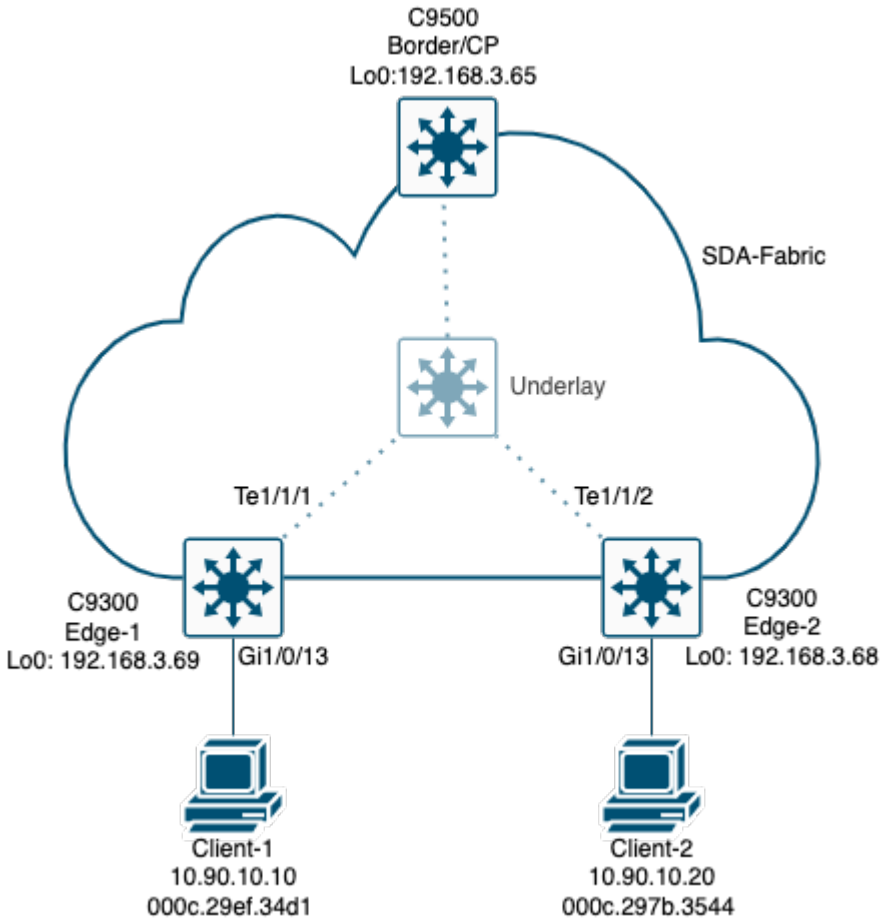
The SD-Access architecture is supported by fabric technology implemented for the campus. It enables the use of virtual networks (overlay networks) that run on the top of a physical network (underlay network) in order to create alternative topologies to connect devices. For more information about the different components of the Cisco SD-Access Solution, please visit:

Cisco SD-Access Solution Design Guide

Layer-2 in the Fabric

- Full Ethernet frame gets encapsulated in VxLAN and transported through SDA Fabric.
- Layer-2 traffic across the Fabric is sent via the Layer-2 LISP Instance.
- Original Layer 2 header is preserved inside the VxLAN encapsulated packet.
- MAC Addresses are registered with Control-Plane (CP) node as EIDs (Endpoint IDs) attached to an specific RLOC (Routing Locator, for example an Edge node).
- Edge Nodes resolve and cache remote MAC addresses.

Network Diagram



Verify

L2-LISP Instance Scale

The actual usable number of *L2-LISP instances* is *64 less* than the max number on the SDM template:

<#root>

```
EDGE-1#
```

show plat hardware fed switch active fwd-asic resource tcam utilization

```
CAM Utilization for ASIC  [0]
 Table                                              Max Values        Used Values
 ------------------------------------------------------------------------------
 Unicast MAC addresses                             32768/1024         44/21
 L3 Multicast entries                               8192/512           4/10
 L2 Multicast entries                               8192/512           1/9
 Directly or indirectly connected routes           24576/8192         33/81
 QoS Access Control Entries                         5120              153
 Security Access Control Entries                    5120              180
 Ingress Netflow ACEs                                256               8
 Policy Based Routing ACEs                          1024              20
 Egress Netflow ACEs                                 768               8
 Flow SPAN ACEs                                     1024              13
 Control Plane Entries                               512              255
 Tunnels                                             512               18

 Lisp Instance Mapping Entries                       512               16
```

```
 Input Security Associations                          256             4
 Output Security Associations and Policies            256             5
 SGT_DGT                                             8192/512         0/1
 CLIENT_LE                                           4096/256         0/0
 INPUT_GROUP_LE                                      1024             0
 OUTPUT_GROUP_LE                                     1024             0
 Macsec SPD                                           256             2
```

In this case, the actual usable number of L2-LISP instances for an Edge node loaded with Cisco IOS® XE 16.9.8 is *448* (512 - 64).

Two hosts that reside on same VN (Virtual Network), same VLAN/Subnet, but attached to different Edge switches. Both Edge switches are part of same SDA Fabric cloud as depicted in the topology image. The two hosts *Client-1* and *Client-2* are part of same VN *Campus_VN* which is attached to *VLAN 1021 / Subnet 10.90.10.1/24*. ICMP packets (pings) are used to test connectivity across both hosts.

```
<#root>

Client-1>

ping 10.90.10.20


Pinging 10.90.10.20 with 32 bytes of data:
Reply from 10.90.10.20: bytes=32 time=4ms TTL=128
Reply from 10.90.10.20: bytes=32 time<1ms TTL=128
Reply from 10.90.10.20: bytes=32 time<1ms TTL=128

Ping statistics for 10.90.10.20:

    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 4ms, Average = 1ms


Client-2>

ping 10.90.10.10


Pinging 10.90.10.10 with 32 bytes of data:
Reply from 10.90.10.10: bytes=32 time<1ms TTL=128
Reply from 10.90.10.10: bytes=32 time<1ms TTL=128
Reply from 10.90.10.10: bytes=32 time<1ms TTL=128

Ping statistics for 10.90.10.10:

    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Since both Edge switches are part of same SDA Fabric, then all production traffic between Edge-1 and Edge-2 needs to be VxLAN encapsulated. In this case, the Edge switches use *L3 Instance ID (IID) 4100 and L2 Instance ID 8191* to encapsulate the traffic.

Control-Plane

You first need to confirm that Control-Plane information is correct. If the information from Control-Plane (software state) looks fine, then you need to verify the Data-Plane (hardware states).

Step 1. Proper SVI Provisioning

As previously mentioned, both hosts from our first scenario reside on *VLAN 1021* and this VLAN/Subnet is stretched across the SDA-Fabric. First, you need to check the configuration of the SVI from *VLAN 1021* that was automatically provisioned by *Digital Network Architecture Center* (DNA Center, or DNAC) on each of the Edge switches:

```
<#root>

EDGE-1#

show run int vlan 1021


Building configuration...

Current configuration : 618 bytes
!
interface Vlan1021

 description Configured from Cisco DNA-Center
 mac-address 0000.0c9f.f55e
 vrf forwarding Campus_VN
 ip address 10.90.10.1 255.255.255.0
 ip helper-address 10.122.150.179

 no ip redirects
 ip route-cache same-interface
 no lisp mobility liveness test
 lisp mobility CAMPUS-WIRED-IPV4
end

EDGE-2#

show run int vlan 1021

Building configuration...

Current configuration : 618 bytes
!
interface Vlan1021

 description Configured from Cisco DNA-Center
 mac-address 0000.0c9f.f55e
 vrf forwarding Campus_VN
 ip address 10.90.10.1 255.255.255.0

 ip helper-address 10.122.150.179
 no ip redirects
 ip route-cache same-interface
 no lisp mobility liveness test
 lisp mobility CAMPUS-WIRED-IPV4
end
```

As you can see in this output, neither L2 nor L3 Instance IDs (IID) are part of the SVI configuration. On an SDA environment, these Instances are automatically configured by DNAC. Therefore, in order to find this information, you have to check the device LISP *running-configuration*. However, if you have hundreds of VLANs, it is not an easy task to find the info of the actual VLAN you want to verify

> **Tip**: Tip: If you do not know the L2 IID information beforehand, you can run this command to find it (use the 'include' filter for the VLAN in question, in our case *VLAN 1021*

EDGE-1#show lisp instance-id * ethernet database | include Vlan 1021 LISP ETR MAC Mapping Database for EID-table *Vlan 1021 (IID 8191)*, LSBs: 0x1

Step 2. MAC address Table

First you need to confirm that both MAC addresses (local and remote) are present in the MAC address table of the Edge switches. For the MAC of the local host, you must also have an ARP entry. You need to check same info on both devices.

<#root>

```
EDGE-1#
sh mac address-table | in 1021
1021    0000.0c9f.f55e    STATIC      Vl1021
1021000c.29ef.34d1    DYNAMIC     Gi1/0/13  <<<< Local host

1021    2cab.eb4f.e6f5    STATIC      Vl1021
1021000c.297b.3544    CP_LEARN    Tu0       <<<< Remote host

EDGE-2#
sh mac address-table | in 1021
1021    0000.0c9f.f55e    STATIC      Vl1021
1021    000c.297b.3544    STATIC      Gi1/0/13  <<<< Local host

1021    70d3.79be.9675    STATIC      Vl1021
1021    000c.29ef.34d1    CP_LEARN    Tu0       <<<< Remote host
```

As you can see in this output, there is an entry with *CP_LEARN* as the address type for the remote host. This entry comes from *Tu0*, which is discussed in greater detail during the 'decapsulation' section. It shows *CP_LEARN* because L2-forwarding got this info from LISP Control-Plane (CP).

Step 3. ARP Table

ARP table only includes an entry for the local host, as the remote host location is resolved via LISP and not

directly via ARP:

```
<#root>

EDGE-1#

sh ip arp vrf Campus_VN 10.90.10.10

Protocol  Address          Age (min)  Hardware Addr   Type   Interface

Internet  10.90.10.10             0   000c.29ef.34d1  ARPA   Vlan1021    <<<< Local host

EDGE-1#

sh ip arp vrf Campus_VN 10.90.10.20


EDGE-1# <<<< Empty for remote host


EDGE-2#

sh ip arp vrf Campus_VN 10.90.10.10


EDGE-2# <<<< Empty for remote host


EDGE-2#

sh ip arp vrf Campus_VN 10.90.10.20

Protocol  Address          Age (min)  Hardware Addr   Type   Interface

Internet  10.90.10.20             0   000c.297b.3544  ARPA Vlan1021      <<<< Local host
```

Step 4. LISP Database for Local Hosts

The information you got from the MAC Address Table is also populated in its hardware-state counterpart, which is the MAC Address Table Manager (MATM). For local hosts, *Switch Integrated Security Features* (SISF, also known as Device-Tracking) snoops the info from the clients and notifies *LISP CP* of *L2-EID (MAC)* and *L2-AR EID (IP/MAC)* information, and this is how *LISP Database* is populated:

```
<#root>

EDGE-1#

show lisp instance-id 8191 ethernet database

LISP ETR MAC Mapping Database for EID-table Vlan 1021 (IID 8191), LSBs: 0x1
Entries total 2, no-route 0, inactive

0000c.29ef.34d1
```

```
/48, dynamic-eid Auto-L2-group-8191, inherited from default locator-set rloc_497d4d09-992e-4eaa-92c8-5c7
  Locator       Pri/Wgt  Source     State

192.168.3.69   10/10   cfg-intf   site-self, reachable
```

```
EDGE-2#

show lisp instance-id 8191 ethernet database

LISP ETR MAC Mapping Database for EID-table Vlan 1021 (IID 8191), LSBs: 0x1
Entries total 1, no-route 0, inactive

0000c.297b.3544

/48, dynamic-eid Auto-L2-group-8191, inherited from default locator-set rloc_ccca08ff-fd0f-42e2-9fbb-a65
  Locator       Pri/Wgt  Source     State

192.168.3.68   10/10   cfg-intf   site-self, reachable
```

Additionally, in the LISP Database you can also verify the address-resolution for the local host. This way you can correlate the L2 information with the L3 data. As previously stated, entries for the remote hosts are not populated in the regular ARP table of the device, therefore this command shows the LISP EID info learnt via ARP *only* for the *local hosts*:

Tip: In this output you can also find the L3 IID associated to that host/subnet: *4100*.

```
<#root>

EDGE-1#

show lisp instance-id 8191 ethernet database address-resolution

LISP ETR Address Resolution for EID-table Vlan 1021 (IID 8191)
(*) -> entry being deleted

Hardware Address       Host Address         L3 InstID

000c.29ef.34d1         10.90.10.10/32            4100
```

```
EDGE-2#

show lisp instance-id 8191 ethernet database address-resolution

LISP ETR Address Resolution for EID-table Vlan 1021 (IID 8191)
(*) -> entry being deleted

Hardware Address       Host Address         L3 InstID

000c.297b.3544         10.90.10.20/32            4100
```

Step 5. RLOC of Remote Gosts with LISP Internet Groper (LIG)

One more additional check is to verify the location of the remote host. you can use LIG command to resolve and identify the RLOC where the remote MAC address is located (in this case, you manually trigger the

LIG via the cli, but when switch needs to forward a frame to an unknown destination MAC, then it automatically signals LISP in order to resolve the location of that unknown MAC):

> **Tip**: Since you want to check L2 connectivity, then you have to use the L2 IID and the MAC address of the remote host as EID for the LIG. On the other hand, if you want to check L3 connectivity, you need to use the L3 IID (in this case *4100*) and the actual IP of the host as EID for the LIG.

<#root>

```
EDGE-1#
lig instance 8191 000c.297b.3544
Mapping information for EID 000c.297b.3544 from 192.168.2.2 with RTT 1 msecs
000c.297b.3544
/48, uptime: 05:32:34, expires: 23:59:59, via map-reply, complete
  Locator       Uptime    State  Pri/Wgt     Encap-IID

192.168.3.68  05:32:34  up      10/10        -         <<<< RLOC of Edge-2


EDGE-2#
lig instance 8191 000c.29ef.34d1
Mapping information for EID 000c.29ef.34d1 from 192.168.2.2 with RTT 1 msecs
000c.29ef.34d1
/48, uptime: 05:33:14, expires: 23:59:59, via map-reply, complete
  Locator       Uptime    State  Pri/Wgt     Encap-IID

192.168.3.69  05:33:14  up      10/10        -         <<<< RLOC of Edge-1
```

Step 6. LISP Map-Cache for Remote Hosts

The RLOC from the remote host is also present in the LISP map-cache table (if you do not see the info from your remote host present on this output, then try to do a LIG for that host first and then check again). For remote hosts, LISP updates the L2-forwarding information of the switch with the info present in the LISP map-cache table, which is the reason why the MAC address of the remote host is displayed in the MAC address table of the switch as learnt over *Tu0* with *CP_LEARN* type:

<#root>

```
EDGE-1#
show lisp instance-id 8191 ethernet map-cache

LISP MAC Mapping Cache for EID-table Vlan 1021 (IID 8191), 1 entries
000c.297b.3544
```

```
/48, uptime: 05:36:05, expires: 23:56:28, via map-reply, complete
  Locator       Uptime    State  Pri/Wgt     Encap-IID

192.168.3.68  05:36:05  up      10/10        -         <<<< RLOC of Edge-2


EDGE-2#
```

show lisp instance-id 8191 ethernet map-cache

```
LISP MAC Mapping Cache for EID-table Vlan 1021 (IID 8191), 1 entries
000c.29ef.34d1/48, uptime: 05:36:17, expires: 23:56:56, via map-reply, complete
  Locator       Uptime    State  Pri/Wgt     Encap-IID

192.168.3.69  05:36:17  up      10/10        -         <<<< RLOC of Edge-1
```

Step 7. Switch Integrated Security Features (SISF) Database (Device-Tracking Database)

SISF is involved in the process of snooping Layer 3 to Layer 2 mappings to facilitate learning of End Points (through DHCP and ARP snooping). In L2-LISP, the Edge nodes forward the traffic purely based upon Layer 2 information. SISF comes into play because Broadcast ARP traffic is not forwarded over the Fabric across Ingress/Egress Tunnel Routers (xTRs - which is another name for the Edge nodes in the SDA architecture). *ARP traffic is tunneled, not flooded, through the Fabric.*

The SISF component of the Edge node registers the ARP resolution information from its local hosts (this info is called Endpoint ID - EID) on the Map Server & Map Resolver (MSMR) functionality of the Control-Plane (CP) nodes. The *CP/MSMR* nodes maintain the mapping database that is populated by all Edge nodes. When a host tries to resolve via ARP Request the IP/MAC binding of a remote host located on a different Edge node, the local Edge node intercepts and caches the Broadcast ARP request, it then snoops the packet and queries the *CP/MSMR* for the IP to MAC binding. Finally, the Edge node rewrites the Broadcast Destination Mac to the Unicast Destination Mac - that got from *CP/MSMR* as response to its query -, encapsulates the Unicast ARP request packet in VxLAN format and sends it over the Fabric to the remote Edge node that contains that Destination.

SISF not only helps snoop the packets, it also keeps the local entries refreshed in the Device-Tracking database by appropriate use of ARP probes.

<#root>

```
EDGE-1#
```

sh device-tracking database vlanid 1021

```
vlanDB has 5 entries for vlan 1021, 2 dynamic
<snip>
    Network Layer Address  Link Layer Address   Interface   vlan   prlvl  age   state      Time left
    ARP 10.90.10.20        000c.297b.3544       Tu0         1021   0005   15s   REACHABLE  234s      <<<
    ARP 10.90.10.1         0000c.29ef.34d1      Gi1/0/13    1021   0005   15s   REACHABLE  300s      <<<

EDGE-1#
```

sh device-tracking database mac

```
 MAC             Interface      vlan prlvl      state          time left  policy
```

```
 <snip>
 000c.29ef.34d1   Gi1/0/13       1021 NO TRUST   MAC-REACHABLE  284s       IPDT_POLICY          <<<< Loc
 000c.297b.3544   Tu0            1021 NO TRUST   MAC-REACHABLE  87s        LISP-DT-GUARD-VLAN   <<<< Rem

EDGE-1#
```

sh device-tracking database address all

```
    Network Layer Address    Link Layer Address Interface  vlan prlvl  age   state     Time left
<snip>
    ARP 10.90.10.20          000c.297b.3544     Tu0        1021  0005    2s  REACHABLE  243s   <<<< Remo
    ARP 10.90.10.10          000c.29ef.34d1     Gi1/0/13   1021  0005    2s  REACHABLE  304s   <<<< Loca

EDGE-2#
```

sh device-tracking database vlanid 1021

```
vlanDB has 5 entries for vlan 1021, 2 dynamic
<snip>
    Network Layer Address              Link Layer Address Interface  vlan  prlvl  age  state      Time
    ARP 10.90.10.20                    000c.297b.3544     Gi1/0/13   1021  0005    2s  REACHABLE  250s
    ARP 10.90.10.10                    000c.29ef.34d1     Tu0        1021  0005    2s  REACHABLE  244s

EDGE-2#
```

sh device-tracking database mac

```
 MAC              Interface      vlan prlvl      state          time left policy
<snip>
```

000c.29ef.34d1   Tu0

```
            1021 NO TRUST   MAC-REACHABLE  187s
```

LISP-DT-GUARD-VLAN   <<<< Remote host (info from the CP node via MSMR query)

000c.297b.3544   Gi1/0/13

```
       1021 NO TRUST   MAC-REACHABLE  239s
```

IPDT_POLICY   <<<< Local host

```
EDGE-2#
```

sh device-tracking database address all

```
 Network Layer Address    Link Layer Address Interface   vlan prlvl  age   state      Time left
<snip>
```

ARP 10.90.10.20

000c.297b.3544   Gi1/0/13

```
    1021  0005   29s  REACHABLE  211s
```

<<<< Local host

```
ARP 10.90.10.10          000c.29ef.34d1     Tu0

         1021  0005  138s  REACHABLE  108s

<<<< Remote host (info from the CP node via MSMR query)
```

Data-Plane (Encapsulation Path)

Once you have verified that Control-Plane information is complete and correct, you can now review the Data-Plane part.

Step 1. Entries on MAC Address Table Manager (MATM)

MATM stands for MAC Address Table Manager and the hardware abstraction of the regular MAC address table.

> **Tip**: The commands on this section relate to the hardware-abstraction layer of the device. This means that, if you deployed your devices in stack configuration, you need to run the command not only for the Active member, but also for the member switch you want to verify (for example, if your host is attached to member 2 of the stack, you also have to use 'switch 2' on the cli). For this article only standalone switches were used, therefore only the info for the Active instance is verified.

<#root>

```
EDGE-1

#

show platform software fed switch active matm macTable vlan 1021

VLAN   MAC                   Type  Seq#   EC_Bi  Flags  machandle           siHandle            riHandle
-------------------------------------------------------------------------------------------------------
<snip>

1021   000c.29ef.34d1         0x1

      1      0      0  0x7f9e1caa4358      0x7f9e1caf3fc8      0x0                0x7f9e1c7b5228

GigabitEthernet1/0/13


1021   000c.297b.3544   0x1000001

      0      0     64

0x7f9e1cded158      0x7f9e1ce092f8      0x7f9e1ce08de8


0x7f9e1c2f4a48

              0       16

RLOC 192.168.3.68 adj_id 23


Total Mac number of addresses:: 1
```

```
Summary:
Total number of secure addresses:: 0
Total number of drop addresses:: 0
Total number of lisp local addresses:: 0
Total number of lisp remote addresses:: 0
*a_time=aging_time(secs)  *e_time=total_elapsed_time(secs)
Type:

MAT_DYNAMIC_ADDR           0x1

  MAT_STATIC_ADDR            0x2  MAT_CPU_ADDR               0x4  MAT_DISCARD_ADDR           0x8
MAT_ALL_VLANS             0x10  MAT_NO_FORWARD            0x20  MAT_IPMULT_ADDR           0x40  MAT_RESY
MAT_DO_NOT_AGE           0x100  MAT_SECURE_ADDR          0x200  MAT_NO_PORT              0x400  MAT_DROP
MAT_DUP_ADDR            0x1000  MAT_NULL_DESTINATION    0x2000  MAT_DOT1X_ADDR          0x4000  MAT_ROUT
MAT_WIRELESS_ADDR      0x10000  MAT_SECURE_CFG_ADDR    0x20000  MAT_OPQ_DATA_PRESENT   0x40000  MAT_WIRE
MAT_DLR_ADDR          0x100000  MAT_MRP_ADDR          0x200000  MAT_MSRP_ADDR         0x400000  MAT_LISP

MAT_LISP_REMOTE_ADDR 0x1000000

  MAT_VPLS_ADDR        0x2000000
```

As seen in the previous output, the bit-map type for the MAC addresses of the remote hosts is *0x1000001*: Bit *0x1000000* indicates a LISP remote entry and bit *0x1* is set for a dynamic entry. The other important values from this table are the *machandle*, the *siHandle* and the *riHandle*, keep this info handy for the next verifications.

Step 2. Print Resource Handle Information for 'machandle'

The *machandle* is used to verify the info programmed on hardware for this object, in this case for the remote MAC address:

<#root>

```
EDGE-1

#

show platform hardware fed switch active fwd-asic abstraction



print-resource-handle 0x7f9e1cded158 1

Handle:0x7f9e1cded158 Res-Type:ASIC_RSC_HASH_TCAM Res-Switch-Num:0 Asic-Num:255 Feature-ID:AL_FID_L2_WIR
priv_ri/priv_si Handle: (nil)Hardware Indices/Handles: handle [ASIC: 0]: 0x7f9e1cded368
Features sharing this resource:Cookie length: 12
7b 29 0c 00 44 35 06 80 07 00 00 00

Detailed Resource Information (ASIC# 0)
---------------------------------------
Number of HTM Entries: 1

Entry 0: (handle 0x7f9e1cded368)

Absolute Index: 4100
Time Stamp: 231

KEY - vlan:6

 mac:0xc297b3544 l3_if:0
```

```
gpn:3401

 epoch:0 static:0 flood_en:0 vlan_lead_wless_flood_en: 0 client_home_asic: 0 learning_peerid 0, learning
MASK - vlan:0 mac:0x0 l3_if:0 gpn:0 epoch:0 static:0 flood_en:0 vlan_lead_wless_flood_en: 0 client_home_
SRC_AD - need_to_learn:0 lrn_v:0 catchall:0 static_mac:0 chain_ptr_v:0 chain_ptr: 0 static_entry_v:0 aut
DST_AD -

si:0xd5

 bridge:0 replicate:0 blk_fwd_o:0 v4_rmac:0 v6_rmac:0 catchall:0 ign_src_lrn:0 port_mask_o:0 afd_cli_f:0


============================================================
```

These are the most important fields from the previous output from *Edge-1*:

1. Key VLAN (*MVID*) = 6
2. Group Port Number (*GPN*) = 3401
3. Station Index (*SI*) = 0xd5

Step 3. Key VLAN (MVID)

The *Key VLAN ID* from previous output must match the hardware MVID value assigned to *VLAN 1021*, you can find that info on this command output:

<#root>

```
EDGE-1#

show platform software fed switch active vlan 1021

VLAN Fed Information



Vlan Id

  IF Id                LE Handle            STP Handle           L3 IF Handle         SVI IF ID

MVID


----------------------------------------------------------------------------------------------------------------

1021

     0x0000000000420010   0x00007f9e1c65d268   0x00007f9e1c65da98   0x00007f9e1c995e18   0x0000000000000

6
```

Confirmed, *MVID* value is equal to 6 for *VLAN 1021!*

Step 4. Group Port Number (GPN)

In order to check the *GPN value* used by the L2-LISP *Tunnel0*, a couple of commands are required. First you have to confirm the hardware Interface ID assigned to the *Tunnel0*:

<#root>

EDGE-1

#

show platform software

dpidb l2lisp 8191

Instance Id:8191

, dpidx:4325400,

vlan:1021

, Parent Interface:

Tunnel0(if_id:64)

or alternatively you can use command:

EDGE-1

#show platform software fed sw active

ifm interfaces l2-lisp

```
Interface                         IF_ID               State
----------------------------------------------------------------------
```

Tunnel0   0x00000040

READY

Now, you can verify the attributes assigned to that interface ID in the Interface Feature Manager (IFM):

<#root>

EDGE-1

#

show platform software fed switch active

ifm if-id 64

<<<< 64 DEC = 0x40 HEX

Interface IF_ID   : 0x0000000000000040

Interface Name   : Tunnel0

```
Interface Block Pointer : 0x7f9e1c91d5c8
Interface Block State   : READY
```

```
Interface State         : Enabled
Interface Status        : ADD, UPD
Interface Ref-Cnt       : 2

Interface Type          : L2_LISP              <<<< Tunnel Type

        Is top interface    : TRUE
        Asic_num            : 0
        Switch_num          : 0
        AAL port Handle     : cc00005d


Source Ip Address   : 192.168.3.69     <<<< Tunnel Source Address (Lo0), RLOC of Edge-1

        Vlan Id             : 0
        Instance Id         : 0


Dest Port           : 4789              <<<< VxLAN UDP Port



SGT                 : Disable           <<<< CTS not configured for this scenario

        Underlay VRF (V4)   : 0
        Underlay VRF (V6)   : 0
        Flood Access-tunnel : Disable
        Flood unknown ucast : Disable
        Broadcast           : Disable
        Multicast Flood     : Disable

Decap Information

                TT HTM handle     : 0x7f9e1c9c40e8

        Port Information
        Handle ............ [0xcc00005d]
        Type .............. [L2-LISP-top]
        Identifier ........ [0x40]
        Unit .............. [64]
         L2 LISP Topology interface Subblock
                Switch Num        : 1
                Asic Num          : 0
                Encap PORT LE handle    : 0x7f9e1c9befe8
                Decap PORT LE handle    : 0x7f9e1c91d818
                L3IF LE handle    : 0x7f9e1c9bf2b8
                SI handle decap   : 0x7f9e1c9c8568
                DI handle         : 0x7f9e1c2f4a48
                RI handle         : 0x7f9e1c9c4498
                RCP Service ID    : 0x0


GPN               : 3401         <<<< GPN

                TRANS CATCH ALL handle : 0x7f9e1c2f5698
<snip>
```

Apart from the *GPN* value of *3401*, which matches the info obtained during the hardware abstraction verification of the *machandle* for the remote MAC, in the previous output you have some other useful information that is used to encapsulate the traffic when is sent over the L2-LISP tunnel, like for example: Tunnel Type, Tunnel Source IP Address, UDP Destination Port, and so on.

Step 5. Station Index (SI)

With the *station index* you can obtain the destination index and rewrite index used to send the traffic over the L2-LISP tunnel. This info tells us *HOW* and *WHERE* to send the packets:

> **Tip**: At this step you collect two more additional values, the DI (destination index) and the RI (rewrite index), keep them handy for future reference.

<#root>

EDGE-1

#

show platform hardware fed switch active fwd-asic resource asic all

station-index range 0xd5 0xd5

ASIC#0:
Station Index (SI) [0xd5]
RI = 0x28 <<<< Rewrite Index
DI = 0x5012 <<<< Destination Index

```
stationTableGenericLabel = 0
stationFdConstructionLabel = 0x7
lookupSkipIdIndex = 0
rcpServiceId = 0
dejaVuPreCheckEn = 0
Replication Bitmap: LD
```

ASIC#1:
Station Index (SI) [0xd5]
RI = 0x28
DI = 0x5012

```
stationTableGenericLabel = 0
stationFdConstructionLabel = 0x7
lookupSkipIdIndex = 0
rcpServiceId = 0
dejaVuPreCheckEn = 0
Replication Bitmap: RD CD
```

Step 6. Destination Index (DI)

The important takeaway from this section is that the port-map value (*pmap*) for the physical ports is all-zeroes and the recirculation port-map (*rcp_pmap*) is equal to one on ASIC 0. Since these values use a boolean logic, then this output actually means that the switch does not use a physical port, but a logical interface - *Tunnel0* -, to forward the traffic. Note that the *rcp_pmap* is ON only for ASIC 0.

> **Tip**: The actual ASIC used to forward the traffic depends on the physical port(s) used to establish the L2-LISP tunnel (the underlay connection to the upstream device), as each physical port is mapped to an specific ASIC instance. Also consider that the number of ASICs on the switch varies for each model.

<#root>

EDGE-1

#

show platform hardware fed switch active fwd-asic resource asic all

destination-index range 0x5012 0x5012

ASIC#0:

Destination Index (DI) [0x5012]
portMap = 0x00000000 00000000   <<<< All bits for physical ports are off

cmi1 = 0

rcpPortMap = 0x1   <<<< Recirculation port-map bit is enabled

CPU Map Index (CMI) [0]
ctiLo0 = 0
ctiLo1 = 0
ctiLo2 = 0
cpuQNum0 = 0
cpuQNum1 = 0
cpuQNum2 = 0
npuIndex = 0
stripSeg = 0
copySeg = 0

ASIC#1:
Destination Index (DI) [0x5012]
portMap = 0x00000000 00000000   <<<< All bits for physical ports are off

cmi1 = 0
rcpPortMap = 0

CPU Map Index (CMI) [0]
ctiLo0 = 0
ctiLo1 = 0
ctiLo2 = 0
cpuQNum0 = 0
cpuQNum1 = 0
cpuQNum2 = 0
npuIndex = 0
stripSeg = 0
copySeg = 0

Step 7. Rewrite Index (RI)

In order to verify this index, you have to use the *riHandle* associated to the remote MAC on the MATM table output from *Step 1*, which in this case is *0x7f9e1ce08de8*. The Rewrite Index provides the final details of the VxLAN header that is imposed to the packet before it is sent over the L2-LISP tunnel:

Tip: RI value *40* from this output must match RI index *0x28* from Step 5 *(40 DEC = 0x28 HEX)*.

<#root>

EDGE-1

#

show platform hardware fed switch active fwd-asic abstraction

print-resource-handle 0x7f9e1ce08de8 1

```
Handle:0x7f9e1ce08de8 Res-Type:ASIC_RSC_RI Res-Switch-Num:255 Asic-Num:255 Feature-ID:AL_FID_L2_WIRELESS
priv_ri/priv_si Handle: 0x7f9e1cded678Hardware Indices/Handles: index0:0x28  mtu_index/l3u_ri_index0:0x0
Features sharing this resource:58 (1)]
Cookie length: 56
00 00 00 00 00 00 00 00 fd 03 00 00 00 00 00 00 00 00 00 00 07 00 00 0c 29 7b 35 44 00 00 00 00 00 00 00
```

Detailed Resource Information (ASIC# 0)

```
----------------------------------------
```

Rewrite Data Table Entry,

ASIC#:0, rewrite_type:116, RI:40 <<<< Must match RI Index 0x28 from Step 5

Src IP: 192.168.3.69 <<<< VxLAN header (RLOC of the Local Edge node Edge-1)
Dst IP: 192.168.3.68 <<<< VxLAN header (RLOC of the Remote Edge node Edge-2)
iVxlan dstMac: 0x0c:0x297b:0x3544 <<<< MAC address of the Remote host

```
iVxlan srcMac:  0x00:0x00:0x00
IPv4 TTL:       0
iid present:    1
lisp iid:       0
lisp flags:     0
dst Port:       46354
update only l3if:       0
is Sgt:         1
is TTL Prop:    0
L3if LE:        0 (0)
Port LE:        276 (0)
Vlan LE:        6 (0)
```

Detailed Resource Information (ASIC# 1)

```
----------------------------------------
```

Rewrite Data Table Entry,

ASIC#:1, rewrite_type:116, RI:40

Src IP: 192.168.3.69
Dst IP: 192.168.3.68
iVxlan dstMac: 0x0c:0x297b:0x3544

```
 iVxlan srcMac:  0x00:0x00:0x00
 IPv4 TTL:       0
 iid present:    1
 lisp iid:       0
 lisp flags:     0
 dst Port:       46354
 update only l3if:       0
 is Sgt:         1
 is TTL Prop:    0
 L3if LE:        0 (0)
 Port LE:        276 (0)
 Vlan LE:        6 (0)

==============================================================
```

Step 8. Forwarding Decision for the New Encapsulated Packet

The forwarding decision for the original ICMP Packet directed to *Client-2 - IP:10.90.10.20* points to the LISP interface:

<#root>

EDGE-1#

sh ip cef vrf Campus_VN 10.90.10.20

10.90.10.0/24

attached to LISP0.4100

Once the original packet has been encapsulated and the proper VxLAN headers have been appended to it, you now have to check the forwarding decision based on the top VxLAN fields. In this case the destination IP address *192.168.3.68* which is the RLOC from remote Edge-2 switch:

<#root>

EDGE-1#

show ip

route 192.168.3.68

```
Routing entry for 192.168.3.68/32
  Known via "isis", distance 115, metric 30, type level-1
  Redistributing via isis
  Advertised by isis (self originated)
  Last update from 192.168.3.74 on TenGigabitEthernet1/1/1, 01:15:14 ago
  Routing Descriptor Blocks:
```

*** 192.168.3.74, from 192.168.3.68, 01:15:14 ago, via TenGigabitEthernet1/1/1**

```
      Route metric is 30, traffic share count is 1
```

```
EDGE-1#
```

show ip

cef 192.168.3.68 detail

```
192.168.3.68/32, epoch 3, per-destination sharing
  Adj source: IP midchain out of Tunnel0, addr 192.168.3.68 7FEADF30A390
    Dependent covered prefix type adjfib, cover 0.0.0.0/0
  1 RR source [no flags]
```

nexthop 192.168.3.74 TenGigabitEthernet1/1/1

```
EDGE-1#
```

show

adjacency 192.168.3.68 detail

```
Protocol Interface                 Address
IP       Tunnel0                   192.168.3.68(4)
                                   0 packets, 0 bytes
                                   epoch 0
                                   sourced in sev-epoch 10
                                   Encap length 28
                                   4500000000000000FF113413C0A80345
                                   C0A8034412B512B500000000
```

Tun endpt **<<<< Adjacency type: Tunnel**

```
                                   Next chain element:
```

IP adj out of TenGigabitEthernet1/1/1, addr 192.168.3.74 <<<< Upst

In order to reach IP *192.168.3.68*, traffic has to go through next-hop *192.168.3.74* over interface *Te1/1/1*, therefore you also need to check the adjacency for the next-hop IP:

<#root>

```
EDGE-1#
```

show ip route 192.168.3.74

```
Routing entry for 192.168.3.74/31
  Known via "connected", distance 0, metric 0 (connected, via interface)
  Advertised by isis level-2
  Routing Descriptor Blocks:
```

*** directly connected, via TenGigabitEthernet1/1/1**

```
      Route metric is 0, traffic share count is 1

EDGE-1#
```

show ip cef 192.168.3.74 detail

192.168.3.74/32, epoch 3, flags [attached]
  Interest List:
     - fib bfd tracking
  BFD state up, tracking attached BFD session on TenGigabitEthernet1/1/1
  Adj source:

IP adj out of TenGigabitEthernet1/1/1, addr 192.168.3.74

 7FEADEADCFA8
    Dependent covered prefix type adjfib, cover 192.168.3.74/31
  1 IPL source [no flags]

attached to TenGigabitEthernet1/1/1

EDGE-1#

show

adjacency 192.168.3.74 detail

Protocol Interface   Address
IP   TenGigabitEthernet1/1/1   192.168.3.74(40)
                                   0 packets, 0 bytes
                                   epoch 0
                                   sourced in sev-epoch 10
                                   Encap length 14

00A3D14415582CABEB4FE6C6

0800

<<<< Layer-2 Rewrite Information for the traffic forwarded through this adjacency

                                   L2 destination address byte offset 0
                                   L2 destination address byte length 6

Link-type after encap: ip
                                   ARP

EDGE-1#

show

interfaces

tenGigabitEthernet 1/1/1

| in bia

  Hardware is Ten Gigabit Ethernet, address is 2cab.eb4f.e6c6 (bia 2cab.eb4f.e6c6)

EDGE-1#

```
show


ip


arp Te1/1/1

Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet

192.168.3.74

           98

00a3.d144.1558

  ARPA   TenGigabitEthernet1/1/1
```

Data-Plane (Decapsulation Path)

Step 1. Decap Information from IFM

In order to verify how an incoming VxLAN packet is handled by the switch, first you need to understand how the traffic is decapsulated when is received on *Tunnel0* virtual interface. Remember the *Interface Manager* (IFM) command you collected on a previous step? Back then, you checked the information on the first portion of the command output, now you have to verify the second part of the command output, the one that relates to *Decap Information*:

```
<#root>

EDGE-1#

show platform software fed switch active ifm if-id 64

Interface IF_ID         : 0x0000000000000040

Interface Name          : Tunnel0

Interface Block Pointer : 0x7f9e1c91d5c8
<snip>

Decap Information


                TT HTM handle     : 0x7f9e1c9c40e8

        Port Information
        Handle ............ [0xcc00005d]


Type .............. [L2-LISP-top]

        Identifier ........ [0x40]
        Unit .............. [64]
         L2 LISP Topology interface Subblock
                Switch Num        : 1
                Asic Num          : 0
                Encap PORT LE handle    : 0x7f9e1c9befe8
```

```
                Decap PORT LE handle    : 0x7f9e1c91d818

L3IF LE handle    : 0x7f9e1c9bf2b8



SI handle decap   : 0x7f9e1c9c8568    <<<< Station Index Handle

                DI handle         : 0x7f9e1c2f4a48

RI handle         : 0x7f9e1c9c4498    <<<< Rewrite Index Handle

                RCP Service ID    : 0x0
                GPN               : 3401
                TRANS CATCH ALL handle : 0x7f9e1c2f5698
        Port L2 Subblock
                Enabled ............. [No]
                Allow dot1q ......... [No]
                Allow native ........ [No]
                Default VLAN ........ [0]
                Allow priority tag ... [No]
                Allow unknown unicast  [No]
                Allow unknown multicast[No]
                Allow unknown broadcast[No]
                Allow unknown multicast[Enabled]
                Allow unknown unicast  [Enabled]
                Protected ............ [No]
                IPv4 ARP snoop ....... [No]
                IPv6 ARP snoop ....... [No]
                Jumbo MTU ............ [0]
                Learning Mode ........ [0]
                Vepa ................. [Disabled]
        Port QoS Subblock
                Trust Type ..................... [0x7]
                Default Value .................. [0]
                Ingress Table Map .............. [0x0]
                Egress Table Map ............... [0x0]
                Queue Map ...................... [0x0]
        Port Netflow Subblock
        Port CTS Subblock
                Disable SGACL .................. [0x0]
                Trust .......................... [0x0]
                Propagate ...................... [0x0]
        %Port SGT ........................ [1251474769]
Ref Count : 2 (feature Ref Counts + 1)
IFM Feature Ref Counts
        FID : 95 (AAL_FEATURE_L2_MULTICAST_IGMP), Ref Count : 1
No Sub Blocks Present
```

You have to take into consideration two values from this output: The *L3 Interface Logical-Entity (L3IF LE) Handle* and the *Station Index (SI) Handle* from the *Decap Information* section.

Step 2. Features Programmed on Tunnel Interface via L3IF LE Handle

In order to verify the features associated to *Tunnel0* interface, you need to pull the resource handle information from the associated *L3IF LE Handle*. In this output you can see the features enabled on that interface in a boolean logic, for example: *LISP_VXLAN_ENABLE_IPV4* feature is enabled on this tunnel

interface.

<#root>

EDGE-1#

show platform hardware fed switch active fwd-asic abstraction print-resource-handle 0x7f9e1c9bf2b8 1 | i

LEAD_L3IF_LISP_VXLAN_ENABLE_IPV4 value 1 Pass   <<<< ASIC 0

LEAD_L3IF_LISP_VXLAN_ENABLE_IPV6 value 0 Pass

LEAD_L3IF_LISP_VXLAN_ENABLE_IPV4 value 1 Pass   <<<< ASIC 1

LEAD_L3IF_LISP_VXLAN_ENABLE_IPV6 value 0 Pass

Step 3. Station Index (SI) Handle for Tunnel Interface (Decap Process)

You need to use the resource handle command one more time, in order to check the *Station Index (SI Handle)* and get the *Rewrite Index (RI)* and *Destination Index (DI)* used by the traffic that is received over the *Tunnel0* interface and that needs to be decapsulated by the switch before it is sent to its final destination via regular layer-2 forwarding (local MAC address table):

<#root>

EDGE-1#

show platform hardware fed switch active fwd-asic abstraction

print-resource-handle 0x7f9e1c9c8568 1

Handle:0x7f9e1c9c8568 Res-Type:ASIC_RSC_SI Res-Switch-Num:255 Asic-Num:255 Feature-ID:AL_FID_LISP Lkp-ft
priv_ri/priv_si Handle: 0x7f9e1c9c4498Hardware Indices/Handles: index0:0xac mtu_index/l3u_ri_index0:0x0
Features sharing this resource:109 (1)]
Cookie length: 56
40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Detailed Resource Information (ASIC# 0)

Station Index (SI) [0xac]
RI = 0xa800   <<<< Rewrite Index
DI = 0x5012   <<<< Destination Index

stationTableGenericLabel = 0
stationFdConstructionLabel = 0x4
lookupSkipIdIndex = 0
rcpServiceId = 0
dejaVuPreCheckEn = 0
Replication Bitmap: LD

Detailed Resource Information (ASIC# 1)

```
---------------------------------------


Station Index (SI) [0xac]
RI = 0xa800
DI = 0x5012

stationTableGenericLabel = 0
stationFdConstructionLabel = 0x4
lookupSkipIdIndex = 0
rcpServiceId = 0
dejaVuPreCheckEn = 0
Replication Bitmap: RD CD
================================================================
```

Step 4. Destination Index (DI) and Rewrite Index (RI) Handle for Tunnel interface (Decap Process)

From previous outputs you already know that *DI = 0x5012* means internal recirculation, which makes sense, because switch needs to internally recirculate the packet to do the deposition of the VxLAN header. This means that when switch gets a VxLAN packet on the Tunnel interface, that packet needs to be recirculated to remove the VxLAN header to be able to deliver it to its final destination with the use of the Destination MAC address from the original Frame. In order to verify the *Rewrite Index*, you have to check the resource handle information from the *RI Handle* collected on *Step 1* of this section:

> **Tip**: RI value *43008* from this output must match RI index 0xa800 from previous step (*43008 DEC = 0xa800 HEX*).

```
<#root>

EDGE-1#

show platform hardware fed switch active fwd-asic abstraction


print-resource-handle 0x7f9e1c9c4498 1


Handle:0x7f9e1c9c4498 Res-Type:ASIC_RSC_PORT_LE_RI Res-Switch-Num:255 Asic-Num:255 Feature-ID:AL_FID_LIS
priv_ri/priv_si Handle: 0x7f9e1c9c87f8Hardware Indices/Handles: index0:0xa800  mtu_index/l3u_ri_index0:0
Features sharing this resource:109 (1)]
Cookie length: 56
40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00


Detailed Resource Information (ASIC# 0)

---------------------------------------


Rewrite Data Table Entry,

ASIC#:0, rewrite_type:114, RI:43008         <<<< 43008 DEC = 0xa800 HEX


 Port LE handle:         0

 Port LE Index:  275
```

```
Detailed Resource Information (ASIC# 1)

---------------------------------------


Rewrite Data Table Entry,

ASIC#:1, rewrite_type:114, RI:43008

 Port LE handle:         0

 Port LE Index:  275


============================================================
```

Troubleshoot

Packet Captures with Embedded Packet Capture (EPC) Tool

You can use EPC tool to confirm that the packets are encapsulated with the correct VxLAN info when forwarded over the *Tunnel0* interface. In order to do this, you just have to set the EPC capture on the physical interfaces that composed *Tunnel0* (your underlay connections to the upstream device) and use a filter to catch only the info that is sent to the RLOC of the other Edge switch:

```
<#root>

EDGE-1#

show


ip


access-lists TAC

Extended IP access list TAC


10 permit ip host 192.168.3.69 host 192.168.3.68
    20 permit ip host 192.168.3.68 host 192.168.3.69


EDGE-1#

mon cap tac int te1/1/1 both access-list TAC buffer size 100

EDGE-1#

show


mon
```

cap

tac

Status Information for Capture tac
  Target Type:

Interface: TenGigabitEthernet1/1/1, Direction: BOTH

   Status : Inactive
  Filter Details:

Access-list: TAC

  Buffer Details:
   Buffer Type: LINEAR (default)

Buffer Size (in MB): 100

  File Details:
   File not associated
  Limit Details:
   Number of Packets to capture: 0 (no limit)
   Packet Capture duration: 0 (no limit)
   Packet Size to capture: 0 (no limit)
   Packet sampling rate: 0 (no sampling)

EDGE-1#

mon

cap

tac start

Started capture point : tac

Four ICMP Requests from local host 10.90.10.10 to remote host 10.90.10.20 were sent and then the c

EDGE-1#

mon

cap

tac stop

Capture statistics collected at software:
        Capture duration - 19 seconds

        Packets received - 12

        Packets dropped - 0

```
        Packets oversized - 0

Bytes dropped in asic - 0

Capture buffer will exists till exported or cleared
```

Stopped capture point : tac

```
EDGE-1#show mon cap tac buffer brief
Starting the packet display ........ Press Ctrl + Shift + 6 to exit

  1   0.000000
```

00:0c:29:ef:34:d1 -> 00:0c:29:7b:35:44 ARP 110 Who has 10.90.10.20? Tell 10.90.10.10   <<<< Unicast ARP

```
  2   0.000744 00:0c:29:7b:35:44 -> 00:0c:29:ef:34:d1 ARP 110 10.90.10.20 is at 00:0c:29:7b:35:44
  3   0.001387
```

10.90.10.10 -> 10.90.10.20 ICMP 124 Echo (ping) request

```
  id=0x0001, seq=66/16896, ttl=128
  4   0.131122
```

00:0c:29:7b:35:44 -> 00:0c:29:ef:34:d1 ARP 110 Who has 10.90.10.10? Tell 10.90.10.20   <<<< Unicast ARP

```
  5   0.132059 00:0c:29:ef:34:d1 -> 00:0c:29:7b:35:44 ARP 110 10.90.10.10 is at 00:0c:29:ef:34:d1
  6   0.299394
```

10.90.10.20 -> 10.90.10.10 ICMP 124 Echo (ping) reply   id=0x0001, seq=66/16896, ttl=128 (request in 3

```
  7   0.875191  10.90.10.10 -> 10.90.10.20  ICMP 124 Echo (ping) request  id=0x0001, seq=67/17152, ttl=1
  8   0.875465  10.90.10.20 -> 10.90.10.10  ICMP 124 Echo (ping) reply    id=0x0001, seq=67/17152, ttl=1
  9   1.889098  10.90.10.10 -> 10.90.10.20  ICMP 124 Echo (ping) request  id=0x0001, seq=68/17408, ttl=1
 10   1.889384  10.90.10.20 -> 10.90.10.10  ICMP 124 Echo (ping) reply    id=0x0001, seq=68/17408, ttl=1
 11   2.902932  10.90.10.10 -> 10.90.10.20  ICMP 124 Echo (ping) request  id=0x0001, seq=69/17664, ttl=1
 12   2.903234  10.90.10.20 -> 10.90.10.10  ICMP 124 Echo (ping) reply    id=0x0001, seq=69/17664, ttl=1
```

You can also see the entire packet details with 'buffer detailed' option (use a filter for the appr

```
EDGE-1#
```

show

mon

cap

tac buffer detailed | be Frame 7

```
Frame 7: 124 bytes on wire (992 bits), 124 bytes captured (992 bits) on interface 0
<snip>
    [Protocols in frame: eth:ethertype:ip:udp:vxlan:eth:ethertype:ip:icmp:data]
```

Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)

```
    Destination: 00:00:00:00:00:00 (00:00:00:00:00:00)
        Address: 00:00:00:00:00:00 (00:00:00:00:00:00)
```

```
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Source: 00:00:00:00:00:00 (00:00:00:00:00:00)
        Address: 00:00:00:00:00:00 (00:00:00:00:00:00)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)

Internet Protocol Version 4, Src: 192.168.3.69, Dst: 192.168.3.68    <<<< Outer IP Data (VxLAN header)

    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        0000 00.. = Differentiated Services Codepoint: Default (0)
        .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 110
    Identification: 0x2204 (8708)
    Flags: 0x02 (Don't Fragment)
        0... .... = Reserved bit: Not set
        .1.. .... = Don't fragment: Set
        ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 255

    Protocol: UDP (17)

    Header checksum: 0xd1a0 [validation disabled]
        [Good: False]
        [Bad: False]

    Source: 192.168.3.69

    Destination: 192.168.3.68

User Datagram Protocol

, Src Port: 65344 (65344),

Dst Port: 4789 (4789)

    Source Port: 65344

    Destination Port: 4789          <<<< VxLAN UDP Port

    Length: 90
    Checksum: 0x0000 (none)
        [Good Checksum: False]
        [Bad Checksum: False]
    [Stream index: 0]

Virtual eXtensible Local Area Network

    Flags: 0x8800, GBP Extension, VXLAN Network ID (VNI)
        1... .... .... .... = GBP Extension: Defined
        .... .... .0.. .... = Don't Learn: False
        .... 1... .... .... = VXLAN Network ID (VNI): True
        .... .... .... 0... = Policy Applied: False
        .000 .000 0.00 .000 = Reserved(R): False
    Group Policy ID: 0

    VXLAN Network Identifier (VNI): 8191        <<<< VNI mapped to L2 Instance ID 8191 for L2-LISP

    Reserved: 0


########## Original Frame starts here (Inner headers) ##########
```

```
Ethernet II, Src: 00:0c:29:ef:34:d1 (00:0c:29:ef:34:d1), Dst: 00:0c:29:7b:35:44 (00:0c:29:7b:35:44)
    Destination: 00:0c:29:7b:35:44 (00:0c:29:7b:35:44)  <<<< MAC of Remote host
        Address: 00:0c:29:7b:35:44 (00:0c:29:7b:35:44)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Source: 00:0c:29:ef:34:d1 (00:0c:29:ef:34:d1)       <<<< MAC of Local host
        Address: 00:0c:29:ef:34:d1 (00:0c:29:ef:34:d1)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 10.90.10.10, Dst: 10.90.10.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
        0000 00.. = Differentiated Services Codepoint: Default (0)
        .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 60
    Identification: 0x30b7 (12471)
    Flags: 0x00
        0... .... = Reserved bit: Not set
        .0.. .... = Don't fragment: Not set
        ..0. .... = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: ICMP (1)
    Header checksum: 0xe138 [validation disabled]
        [Good: False]
        [Bad: False]
    Source: 10.90.10.10          <<<< IP of Local host
    Destination: 10.90.10.20     <<<< IP of Remote host
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x4d18 [correct]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 67 (0x0043)
    Sequence number (LE): 17152 (0x4300)
    Data (32 bytes)
0000  61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70   abcdefghijklmnop
0010  71 72 73 74 75 76 77 61 62 63 64 65 66 67 68 69   qrstuvwabcdefghi
        Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
        [Length: 32]
```

Related Information

- [BRKARC-2020: SD Access: Troubleshooting the Fabric]

- Cisco SD-Access Solution Design Guide