Troubleshoot Mac Address Table Manager on Catalyst 9000 Switches

Contents

Introduction

Prerequisites

- Requirements
- Components Used

Background Information

- Terminology

Configure

Verify

- Dynamic Mac Address Learning in FED MATM
- Dynamic Mac Address Learning in IOS MATM
- Static Mac Address Learning in IOS MATM
- Static Mac Address Learning in FED MATM
- EVPN Mac Address Learning in MATM

Troubleshoot

- Network Connectivity Issue
  - Check IP ARP and Mac Address of the Destination Device
  - Check Egress and Ingress Traffic on the Port Connected to the Destination Device
  - Check the Detailed Output of the Return Traffic
  - Check the Mac Address for SVI 100
  - Possible Connectivity Issue Causes
  - Check MATM Hardware Resources
  - MATM Syslog Errors
  - Potential Fixes

Related Information

Introduction

This document describes how to understand and troubleshoot Mac Address Table Manager on Catalyst 9000 Series Switches.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

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

- Cisco Catalyst 9500 High Performance, 9600 Series Switches on Cisco IOS® XE 16.x & 17.x Software
- Cisco Catalyst 9200, 9300, 9400, 9500 Non-High Performance Series Switches on Cisco IOS® XE 16.x & 17.x Software

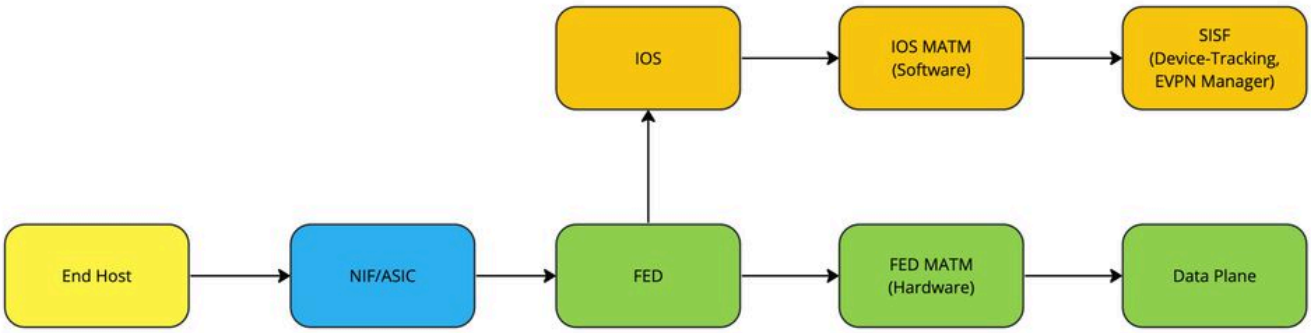
The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Background Information

The Mac Address Table Manager (MATM) is the database where learned Mac Addresses are written and stored. The two types of MATM outlined in this document are:

- IOS MATM (Software)
- FED MATM (Hardware)

When an end-host first sends a packet into a switch, it goes through the NIF/ASIC and get punted into FED. At this point, FED punts this new end-host information up to IOS for IOS MATM to write the information into its database, while simultaneously writing the same information into FED MATM as seen in the diagram:



The importance of each MATM depends on what type of traffic is being passed:

- Locally switch traffic utilizes FED MATM to forward packets in hardware (Data Plane).
- Control Plane functions such as LISP (SDA Fabric) or EVPN that use Device-Tracking Databases, which is built from IOS MATM

Note: When an SVI is created on a switch, it is first created and written in IOS MATM and then pushed down to FED MATM for learning.

Terminology

MATM	Mac Address Table Manager
Mac Address	12-digit unique hardware identifier for a device on a network
diHandle	Destination Index Handle
pmap_intf	Port Map Interface
NIF	Network Interface
FED	Forward Engine Driver
IOS	Internetwork Operating System

Data Plane	Traffic Forwarded on Hardware
SISF	Switch Integrated Security Features
TCAM	Ternary Content-Addressable Memory
SVI	Switch Virtual Interface



Note: Per-platform, the CLI can sometimes include the term switch (however, not always). (show platform soft fed switch <number|active|standby> matm macTable versus show platform soft fed active matm macTable).

Configure

There are no configuration requirements.

Verify

Dynamic Mac Address Learning in FED MATM

<#root>

Switch#

show platform software fed switch active matm macTable vlan 100 mac 9c54.1631.8bd1

```
VLAN   MAC                 Type  Seq#    EC_Bi  Flags  machandle          siHandle           riHandl
-------------------------------------------------------------------------------------------------------
```

100

9c54.1631.8bd1

0x1

248 0 0

0x7f7490c93bd8

0x7f7490c73d98

0x0

0x7f7490a4e108

300 8

TenGigabitEthernet2/1/1

Yes

======platform hardware details ======

Asic: 0

htm-handle = 0x7f7490c80ce8 MVID = 5 gpn = 1
SI = 0xb6 RI = 0x1a DI = 0x537d
DI = 0x537d pmap = 0x00000000 0x00000000

Asic: 1

SI = 0xb6 RI = 0x1a DI = 0x537d

DI = 0x537d pmap = 0x00000000 0x10000000 pmap_intf : [TenGigabitEthernet2/1/1]

This is a snippet from the bottom of the output of

show platform software fed switch active matm macTable

to showcase the classification of

Type

to help indicate how the

Mac Address

is being learned on the

Switch:

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_RES
MAT_DO_NOT_AGE            0x100  MAT_SECURE_ADDR          0x200  MAT_NO_PORT              0x400  MAT_DRO
MAT_DUP_ADDR             0x1000  MAT_NULL_DESTINATION    0x2000  MAT_DOT1X_ADDR          0x4000  MAT_ROU
MAT_WIRELESS_ADDR       0x10000  MAT_SECURE_CFG_ADDR    0x20000  MAT_OPQ_DATA_PRESENT   0x40000  MAT_WIR
MAT_DLR_ADDR           0x100000  MAT_MRP_ADDR          0x200000  MAT_MSRP_ADDR         0x400000  MAT_LIS
MAT_LISP_REMOTE_ADDR 0x1000000  MAT_VPLS_ADDR        0x2000000  MAT_LISP_GW_ADDR     0x4000000
```

Note: Troubleshooting generally begins with checking IOS MATM, however, FED does learn it firs (in this case.)

Dynamic Mac Address Learning in IOS MATM

<#root>

Switch#

show mac address-table address 9c54.1631.8bd1 <--- What IOS Matm sees

```
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
```

100 9c54.1631.8bd1 DYNAMIC Te2/1/1 <--- Showcases which vlan, how its lea

```
Total Mac Addresses for this criterion: 1
```

Check the hardware programming matches the IOS programming for any inconsistencies.

<#root>

Switch#

show platform software fed switch active matm macTable vlan 100 mac 9c54.1631.8bd1 detail

VLAN MAC Type Seq# EC_Bi Flags machandle siHandle riHandl

100

9c54.1631.8bd1

0x1

248 0 0

0x7f7490c93bd8

0x7f7490c73d98

0x0

0x7f7490a4e108

300 5 Yes

Detailed Resource Information (ASIC_INSTANCE# 0)

Number of HTM Entries:

1

Entry 0: (handle 0x7f7490c80ce8)

Absolute Index: 6442
Time Stamp: 5
KEY - vlan:5 mac:0x9c5416318bd1 l3_if:0 gpn:125 epoch:0 static:0 flood_en:0 vlan_lead_wless_flood_en: 0
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 au
DST_AD - si:0xb6 bridge:0 replicate:0 blk_fwd_o:0 v4_rmac:0 v6_rmac:0 catchall:0 ign_src_lrn:0 port_mas

==========
Detailed Resource Information (ASIC_INSTANCE# 0)

Station Index (SI) [0xb6]

```
RI = 0x1a
DI = 0x537d
stationTableGenericLabel = 0
stationFdConstructionLabel = 0x7
lookupSkipIdIndex = 0
rcpServiceId = 0
dejaVuPreCheckEn = 0x1
Replication Bitmap: CD


Detailed Resource Information (ASIC_INSTANCE# 1)
----------------------------------------

Station Index (SI) [0xb6]
RI = 0x1a
DI = 0x537d
stationTableGenericLabel = 0
stationFdConstructionLabel = 0x7
lookupSkipIdIndex = 0
rcpServiceId = 0
dejaVuPreCheckEn = 0x1
Replication Bitmap: LD

============================================================
Detailed Resource Information (ASIC_INSTANCE# 0)
----------------------------------------

Destination index   = 0x537d
pmap                = 0x00000000 0x00000000
cmi                 = 0x0
rcp_pmap            = 0x0
al_rsc_cmi
CPU Map Index (CMI) [0]
ctiLo0              = 0
ctiLo1              = 0
ctiLo2              = 0
cpuQNum0            = 0
cpuQNum1            = 0
cpuQNum2            = 0
npuIndex            = 0
stripSeg            = 0
copySeg             = 0


Detailed Resource Information (ASIC_INSTANCE# 1)          <--- Note the ASIC Instance # as it is based
----------------------------------------

Destination index   = 0x537d
pmap                = 0x00000000 0x10000000
pmap_intf : [TenGigabitEthernet2/1/1]                     <--- Port map interface is learned correctly


cmi                 = 0x0
rcp_pmap            = 0x0
al_rsc_cmi
CPU Map Index (CMI) [0]
ctiLo0              = 0
ctiLo1              = 0
ctiLo2              = 0
cpuQNum0            = 0
cpuQNum1            = 0
cpuQNum2            = 0
```

```
npuIndex            = 0
stripSeg            = 0
copySeg             = 0
============================================================
```

Caution: If you run the detailed command for an interface with the syntax of active, but the interface is on another switch, you do not receive any port map interface output.

Static Mac Address Learning in IOS MATM

This example uses a Switch SVI Mac Address to showcase proper programming:

<#root>

Switch#

show run interface vlan 100 **<--- Verify SVI co**

Building configuration...

Current configuration : 82 bytes
!

interface Vlan100
ip address 192.168.1.2 255.255.255.0

end

Switchk#

show interface vlan 100

Vlan100 is up

, line protocol is up , Autostate Enabled

Hardware is Ethernet SVI, address is 706b.b929.f751

(

bia 706b.b929.f751

)

<--- Mac Address assigned to SVI 100 by the Switch

  Internet address is 192.168.1.2/24
 <snippet>

Switch#

show mac address-table address 706b.b929.f751 **<--- Verify macTa**

          Mac Address Table

Vlan   Mac Address   Type   Ports
----   -----------   --------   -----

100   706b.b929.f751   STATIC   Vl100

Total Mac Addresses for this criterion:

 1

Static Mac Address Learning in FED MATM

<#root>

Switch#

show platform software fed switch active matm macTable vlan 100 **<--- Verify macTab**

VLAN   MAC   Type Seq#   EC_Bi Flags machandle   siHandle   riHandl
----

100   706b.b929.f751   0x8002

       0   0   64 0x7fc210e57908   0x7fc210cb7d78   0x0   0x0

 0   0

Vlan100

                                                Yes

```
100     0027.90be.20d1      0x101     192       0      64  0x7fc210cdc058      0x7fc210cd6da8      0x0


Total Mac number of addresses:: 2
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_RES
MAT_DO_NOT_AGE           0x100  MAT_SECURE_ADDR          0x200  MAT_NO_PORT              0x400  MAT_DRO

 <--- Note 0x8000 + 0x2 == 0x8002 ---> Routed Address that is Statically assigned on the Switch (SVI)

MAT_DUP_ADDR            0x1000  MAT_NULL_DESTINATION    0x2000  MAT_DOT1X_ADDR          0x4000

MAT_ROUTER_ADDR         0x8000

MAT_WIRELESS_ADDR      0x10000  MAT_SECURE_CFG_ADDR    0x20000  MAT_OPQ_DATA_PRESENT   0x40000  MAT_WIR
MAT_DLR_ADDR          0x100000  MAT_MRP_ADDR          0x200000  MAT_MSRP_ADDR         0x400000  MAT_LIS
MAT_LISP_REMOTE_ADDR 0x1000000  MAT_VPLS_ADDR        0x2000000  MAT_LISP_GW_ADDR     0x4000000
```

Note: An SVI created on a switch has no **diHandle** due to it being a **Routed Address**.

EVPN Mac Address Learning in MATM

Determine the Vlan the Mac Address is expected to learn on and verify MATM.

Note: For more information on EVPN, refer to the BGP EVPN VXLAN Configuration Guide.

<#root>

```
Switch#
```

show platform software fed switch active matm macTable vlan 201

```
VLAN   MAC                 Type  Seq#    EC_Bi  Flags  machandle          siHandle           riHandl
-------------------------------------------------------------------------------------------------
201    0006.f601.cd42       0x1   32436      0      0  0x71e058dc3368     0x71e058655018     0x0


201    0006.f601.cd01       0x1   32437      0      0  0x71e058dae308     0x71e058655018     0x0


201

0006.f617.ee81  0x1000001       0      0     64

  0x71e059191ee8      0x71e058e11468

 0x71e058ef0d18

      0x0                         0

5335175  VTEP 172.16.255.4 adj_id 1376

                                 No


Total Mac number of addresses:: 4
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:: 2                        <--- Remotely learned addresses from EV

*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_RES
MAT_DO_NOT_AGE            0x100  MAT_SECURE_ADDR          0x200  MAT_NO_PORT               0x400  MAT_DRO
MAT_DUP_ADDR             0x1000  MAT_NULL_DESTINATION    0x2000  MAT_DOT1X_ADDR           0x4000  MAT_ROU

<--- Note 0x1000000 + 0x1 == 0x1000001 ---> Mac Address remotely learned Dynamically via EVPN

MAT_WIRELESS_ADDR      0x10000  MAT_SECURE_CFG_ADDR    0x20000  MAT_OPQ_DATA_PRESENT   0x40000  MAT_WIR
MAT_DLR_ADDR          0x100000  MAT_MRP_ADDR          0x200000  MAT_MSRP_ADDR          0x400000  MAT_LIS

MAT_LISP_REMOTE_ADDR 0x1000000

  MAT_VPLS_ADDR        0x2000000  MAT_LISP_GW_ADDR     0x4000000
```

Note: The EVPN Type flag uses the same notation of MAT_LISP_REMOTE_ADDR as LISP Mac Learning.

Troubleshoot

Network Connectivity Issue

This example uses a **Switch Stack of 2 C9300-48UN** wherein **SVI 100** is the **L3 Gateway** on the network and its own Mac Address is not programmed correctly including:

- Destination Device is connected to a port on Switch 2
- Source Device is connected to a port on Switch 1
- SVI 100 is a Gateway
- No connectivity to Destination Device from Source Device (ICMP is used to test)
- If Destination Device connects to Switch 1, the connectivity is restored

Check IP ARP and Mac Address of the Destination Device

Switch#**show ip arp 192.168.1.3** Protocol Address Age (min) Hardware Addr Type Interface Internet **192.168.1.3** 6 **9c54.1631.8bd1** ARPA **Vlan100 <--- ARP Resolved on Vlan 100 correctly**
Switch#**show mac add address 9c54.1631.8bd1** Mac Address Table --
Vlan Mac Address Type Ports ---- ----------- -------- ----- **100 9c54.1631.8bd1 DYNAMIC Te2/1/1 <--- IOS Programmed Destination Mac Address correctly** Total Mac Addresses for this criterion: 1

Check Egress and Ingress Traffic on the Port Connected to the Destination Device

The main goal is to ensure the ingress traffic is seen. This can be done via **EPC** and this example uses **ICMP Traffic**.

Switch#**monitor capture tac interface Te2/1/1 both match any start**
<wait some time>

Switch#**monitor capture tac stop**
Switch#**show monitor capture tac buffer brief | i ICMP**
908 4.969635 192.168.1.2 -> 192.168.1.3 ICMP 114 Echo (ping) request id=0x0008, seq=0/0, ttl=255 **909 4.970165 192.168.1.3 -> 192.168.1.2 ICMP 118 Echo (ping) reply id=0x0008, seq=0/0, ttl=254 (request in 908). <--- Replies are seen ingress on EPC showcasing traffic makes it to Destination Device and it replies correctly** 910 4.970425 192.168.1.2 -> 192.168.1.3 ICMP 114 Echo (ping) request id=0x0008, seq=1/256, ttl=255 **911 4.970724 192.168.1.3 -> 192.168.1.2 ICMP 118 Echo (ping) reply id=0x0008,**

seq=1/256, ttl=254 (request in 910) 912 4.970889 192.168.1.2 -> 192.168.1.3 ICMP 114 Echo (ping) request id=0x0008, seq=2/512, ttl=255 **913 4.971211 192.168.1.3 -> 192.168.1.2 ICMP 118 Echo (ping) reply id=0x0008, seq=2/512, ttl=254 (request in 912)** 914 4.971436 192.168.1.2 -> 192.168.1.3 ICMP 114 Echo (ping) request id=0x0008, seq=3/768, ttl=255 **915 4.971558 192.168.1.3 -> 192.168.1.2 ICMP 118 Echo (ping) reply id=0x0008, seq=3/768, ttl=254 (request in 914)**

Check the Detailed Output of the Return Traffic

Switch#**show monitor capture tac buffer detailed | begin Frame 909** Starting the packet display ..... Press Ctrl + Shift + 6 to exit Frame 1: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface /tmp/epc_ws/wif_to_ts_pipe, id 0 Interface id: 0 (/tmp/epc_ws/wif_to_ts_pipe) Interface name: /tmp/epc_ws/wif_to_ts_pipe Encapsulation type: Ethernet (1) Arrival Time: Apr 19, 2024 19:14:13.044770000 UTC [Time shift for this packet: 0.000000000 seconds] Epoch Time: 1713554053.044770000 seconds [Time delta from previous captured frame: 0.000000000 seconds] [Time delta from previous displayed frame: 0.000000000 seconds] [Time since reference or first frame: 0.000000000 seconds] Frame Number: 1 Frame Length: 118 bytes (944 bits) Capture Length: 118 bytes (944 bits) [Frame is marked: False] [Frame is ignored: False] [Protocols in frame: eth:ethertype:vlan:ethertype:ip:icmp:data] **Ethernet II, Src: 9c:54:16:31:8b:d1 (9c:54:16:31:8b:d1), Dst: 70:6b:b9:29:f7:51 (70:6b:b9:29:f7:51) <--- Verify this source mac matches the outputs above from the arp entry / macTable for 192.168.1.3 Destination: 70:6b:b9:29:f7:51 (70:6b:b9:29:f7:51) <--- Verify if this DMAC (Destination Mac Address) matches the Gateway (SVI 100) Address: 70:6b:b9:29:f7:51 (70:6b:b9:29:f7:51)** .... ..0. .... = LG bit: Globally unique address (factory default) ....0 .... = IG bit: Individual address (unicast) Source: 9c:54:16:31:8b:d1 (9c:54:16:31:8b:d1) Address: 9c:54:16:31:8b:d1 (9c:54:16:31:8b:d1) .... ..0. .... = LG bit: Globally unique address (factory default) ....0 .... = IG bit: Individual address (unicast)

Switch#**show interface vlan 100** Vlan100 is up, line protocol is up , Autostate Enabled **Hardware is Ethernet SVI, address is 706b.b929.f751 (bia 706b.b929.f751) <--- Matches DMAC of ICMP Reply** Internet address is 192.168.1.4/24

Check the Mac Address for SVI 100

Switch#**show mac address-table address 706b.b929.f751** Mac Address Table ---------- ---------- Vlan Mac Address Type Ports ---- ----------- -------- ----- <**--- IOS does not have SVI 100 Mac Address programmed at all (unexpected)**

Switch#**show platform software fed switch active matm macTable vlan 100 mac 706b.b929.f751** VLAN MAC Type Seq# EC_Bi Flags machandle siHandle riHandle diHandle *a_time *e_time ports Con ---------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- **100 706b.b929.f751 0x8002 0 0 64 0x7fc210e57908 0x7fc210cb7d78 0x0 0x0 0 0 Vlan100 Yes** ======platform hardware details ====== Asic: 0 <**--- Matm on Switch 1 (Active) has the SVI 100 MAC Programmed Correctly** htm-handle = 0x7fc210cb9e68 MVID = 5 gpn = 1 SI = 0x2d RI = 0x1 DI = 0x5234 Asic: 1 SI = 0x2d RI = 0x1 DI = 0x5234



Note: The reason connectivity is restored when the Destination Device connects to Switch 1 is that MATM is still programmed properly compared to Switch 2.

Switch#**show platform software fed switch 2 matm macTable vlan 100 mac 706b.b929.f751 Total Mac number of addresses:: 0 <--- Matm on Switch 2 does not have the SVI 100 MAC Programmed** 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

Possible Connectivity Issue Causes

Check MATM Hardware Resources

If Switch Hardware Resources, which is responsible for programming Mac Addresses is exhausted, then no more Addresses can be learned

Switch#**show platform hardware fed switch active fwd-asic resource tcam utilization** Codes: EM - Exact_Match, I - Input, O - Output, IO - Input & Output, NA - Not Applicable CAM Utilization for ASIC [0] Table Subtype Dir **Max Used %Used** V4 V6 MPLS Other ---------- ---------- **Mac Address Table EM I 32768 31788 97.01% 0 0 0 31788 <--- Look for 95% or greater values Mac Address Table TCAM I 1024 1019 99.51% 0 0 0 1019** L3 Multicast EM I 8192 0 0.00% 0 0 0 0 L3 Multicast TCAM I 512 9 1.76% 3 6 0 0 L2 Multicast EM I 8192 0 0.00% 0 0 0 0 L2 Multicast TCAM I 512 11 2.15% 3 8 0 0 IP Route Table EM I 24576 3 0.01% 2 0 1 0 IP Route Table TCAM I 8192 19 0.23% 6 10 2 1 QOS ACL TCAM IO 5120 85 1.66% 28 38 0 19 TCAM I 45 0.88% 15 20 0 10 TCAM O 40 0.78% 13 18 0 9 Security ACL TCAM IO 5120 131 2.56% 26 60 0 45 TCAM I 88 1.72% 12 36 0 40 TCAM O 43 0.84% 14 24 0 5 Netflow ACL TCAM I 256 6 2.34% 2 2 0 2 PBR ACL TCAM I 1024 36 3.52% 30 6 0 0 Netflow ACL TCAM O 768 6 0.78% 2 2 0 2 Flow SPAN ACL TCAM IO 1024 13 1.27% 3 6 0 4 TCAM I 5 0.49% 1 2 0 2 TCAM O 8 0.78% 2 4 0 2 Control Plane TCAM I 512 290 56.64% 138 106 0 46 Tunnel Termination TCAM I 512 20 3.91% 8 12 0 0 Lisp Inst Mapping TCAM I 2048 1 0.05% 0 0 0 1 Security Association TCAM I 256 4 1.56% 2 2 0 0 CTS Cell Matrix/VPN Label EM O 8192 0 0.00% 0 0 0 0 CTS Cell Matrix/VPN Label TCAM O 512 1 0.20% 0 0 0 1 Client Table EM I 4096 0 0.00% 0 0 0 0 Client Table TCAM I 256 0 0.00% 0 0 0 0 Input Group LE TCAM I 1024 0 0.00% 0 0 0 0 Output Group LE TCAM O 1024 0 0.00% 0 0 0 0 Macsec SPD TCAM I 256 2 0.78% 0 0 0 2

Note: For more information on Hardware Resources, refer to the Understand Hardware Resources on Catalyst 9000 Switches Guide.

MATM Syslog Errors

MATM Log Message	Definition	Recovery Action
MATM-3-MAX_ENTRIES: Switch 1 F0/0: fed: The maximum	Hardware reserved for Mac Addresses has run out of space	Reduce the scale number of Mac Addresses being learned on the

number of MAC addresses has been reached:32768		Switch

Potential Fixes

Option #1:

Reduce the amount of Mac Addresses learned on the Switch:

- A network loop can be occurring and once that is resolved, Hardware Resources subsides and normal Mac Learning continues
- Network scaling has an effect and use a Switch with larger hardware capacity. (Example: C9300 has maximum of 32768 Mac Address vs C9500H has maximum of 82,000)

Option #2:

A legitimate mis-programming is occurring:

1. Collect all pertinent data:
    - Show tech-support
    - Tracelog Archive
    - Debug MATM
2. **Power Cycle** the device
    - If the issue remains, open a case with Cisco TAC.

Related Information

- Understand Hardware Resources on Catalyst 9000 Switches
- BGP EVPN VXLAN Configuration Guide