Configure L2 Extension for Public Cloud

This chapter describes how to enable enterprise and cloud providers to configure an L2 extension for public clouds with Cisco Catalyst 8000V instances using LISP. Use the command-line interface to extend a layer 2 domain between your public cloud network and the enterprise network.

The following are some of the terminologies and concepts that you should be aware before you configure the LISP Layer 2 Extension:

- **Locator/ID Separation Protocol (LISP)**: LISP is a network architecture and protocol that implements the use of two namespaces instead of a single IP address:
  - Endpoint identifiers (EIDs) - assigned to end hosts.
  - Routing locators (RLOCs) - assigned to devices (primarily routers) that make up the global routing system.
- **LISP-enabled virtualized router**: A virtual machine or appliance that supports routing and LISP functions, including host mobility.
- **Endpoint ID (EID)**: An EID is an IPv4 or IPv6 address used in the source and destination address fields of the first (most inner) LISP header of a packet.
- **Routing locator (RLOC)**: The IPv4 or IPv6 addresses that are used to encapsulate and transport the flow between the LISP nodes. An RLOC is the output of an EID-to-RLOC mapping lookup.
- **Egress Tunnel Router (ETR)**: An ETR is a device that is the tunnel endpoint and connects a site to the LISP-capable part of a core network (such as the Internet), publishes EID-to-RLOC mappings for the site, responds to Map-Request messages, and decapsulates and delivers LISP-encapsulated user data to the end systems at the site. During operation, an ETR sends periodic Map-Register messages to all its configured map servers. These Map-Register messages contain all the EID-to-RLOC entries for the EID-numbered networks that are connected to the ETR's site.
- **Ingress Tunnel Router (ITR)**: An ITR is a device that is the tunnel start point. An ITR is responsible for finding EID-to-RLOC mappings for all traffic destined for LISP-capable sites. When the ITR receives a packet destined for an EID, it first looks for the EID in its mapping cache. If the ITR finds a match, it encapsulates the packet inside a LISP header with one of its RLOCs as the IP source address and one of the RLOCs from the mapping cache entry as the IP destination. The ITR then routes the packet normally.
- **xTR**: A generic name for a device performing both Ingress Tunnel Router (ITR) and Egress Tunnel Router (ETR) functions.

- **PxTR**: The point of interconnection between an IP network and a LISP network, playing the role of ITR and ETR at this peering point.
- **Map-Server (MS)**: An MS is a LISP Infrastructure device that LISP site ETRs register to with their EID prefixes. An MS implements part of the distributed LISP mapping database by accepting registration requests from its client egress tunnel routers (ETRs), aggregating the successfully registered EID prefixes of those ETRs, and advertising the aggregated prefixes into the alternative logical topology (ALT) with border gateway protocol (BGP).

  In a small private mapping system deployment, an MS may be configured to stand alone (or there may be several MSs) with all ETRs configured to register to each MS. If more than one, all MSs have full knowledge of the mapping system in a private deployment.

  In a larger or public mapping system deployment, an MS is configured with a partial mesh of generic routing encapsulation (GRE) tunnels and BGP sessions to other map server systems
- **Map-Resolver (MR)**: An MR is a LISP Infrastructure device to which the ITRs send LISP Map-Request queries when resolving EID-to-RLOC mappings. MRs receive the request and select the appropriate map server

For detailed overview information on LISP and the terminologies, see Locator ID Separation Protocol Overview.

Configure LISP Layer 2 Extension

You can deploy Cisco Catalyst 8000V on public, private, and hybrid clouds. When enterprises move to a hybrid cloud, they need to migrate the servers to the cloud without making any changes to the servers. Enterprises may want to use the same server IP address, subnet mask, and default gateway configurations. They might want to use their own IP addressing scheme in the cloud, and not be limited by the addressing scheme of the cloud provider infrastructure.

To fulfill this requirement, Cisco offers the LISP Layer 2 Extension to Cisco Catalyst 8000V running on Amazon Web Services (AWS), where the Cisco Catalyst 8000V instance acts as the bridge between the enterprise data center and the public cloud. By configuring the LISP Layer 2 Extension, you can extend your Layer 2 networks in the private data center to a public cloud to achieve host reachability between your site and the public cloud. You can also enable the migration of your application workload between the data center and the public cloud.

Benefits

- Carry out data migration with ease and optimize the workload IP address or the firewall rules in your network. Thereby, you can ensure subnet continuity with no broadcast domain extension.
- Virtually add a VM that is on the provider site to facilitate leverage cloud bursting to virtually insert a VM in the Enterprise server while the VM runs on the provider site.

- Provide backup services for partial disaster recovery and disaster avoidance.

Prerequisites for configuring LISP Layer 2 Extension

You must configure each Cisco Catalyst 8000V routerwith one external IP address. In this case, an IPsec tunnel is built between the IP addresses of the two Cisco Catalyst 8000V intances, and the IPsec tunnel has a private address.

Restrictions for configuring LISP Layer 2 Extension

- Enterprise VRF number and VM address numbers are limited on an AWS ECS subnet.
- IPv6 address format is not supported in a Cisco Catalyst 8000V Amazon Machine Image (AMI).

Configure LISP Layer 2 Extension

To configure the L2 extension functionality, you must first deploy the Cisco Catalyst 8000V instance on AWS and configure the instance as an xTR. You must then configure the mapping system to complete the deployment.

The LISP site uses the Cisco Catalyst 8000V instance configured as both an ITR and an ETR (also known as an xTR) with two connections to upstream providers. The LISP site then registers to the standalone device that you have configured as the map resolver/map server (MR/MS) in the network core. The mapping system performs LISP encapsulation and de-encapsulation of the packets that are going to the migrated public IPs. Optionally, for traffic that is leaving AWS, whenever a route to the destination is not found on the routing table, the Cisco Catalyst 8000V instance routes that traffic through the PxTR at the enterprise data center.

Perform the following steps to enable and configure the LISP xTR functionality when using a LISP map server and map resolver for mapping services:

Creating a Cisco Catalyst 8000V instance on AWS

Step 1 Log in to Amazon Web Services. In the left navigation pane, click **VPC**.

Step 2 Click the Start VPC Wizard, and select **VPC with Single Public Subnet** from the left pane.

Step 3 Click **Select**.

Step 4 Create the subnet in the Virtual Private Cloud. Use the following properties:

a) Default Subnet-10.0.0.0/24 (mapped to public IP).

b) Additional subnets-0.0.1.0/24 and 1.0.0.2.0/24. These are private IP addresses and might be internal for the Cisco Catalyst 8000V instance.

Step 5 Select **Create VPC**.

Step 6 Select **Security** > **Network ACLs**.

Step 7 Click **Create Security Group** to create a security group for the Cisco Catalyst 8000V instance. Configure the following properties:

a) Name-SSH-Access

b) TCP Port 22 traffic-Permitted inbound
c) SSH access to C8000V for management-Enabled

Step 8 To create additional security groups, perform step 6.

Step 9 Go to the Cisco Catalyst 8000V product page, and click **Continue**.

Step 10 Click **Launch with E2 Console** to launch the Cisco Catalyst 8000V in accordance with your geographical region.

Step 11 Choose the appropriate instance type. Refer tables 2-1 and 2-2 for supported instance types.

The minimum memory requirement for a medium instance type (m1.medium) is 10Mbps; large instance type (m1.large) is 50Mbps.

ECU stands for Elastic Compute Unit. ECU is Amazon's proprietary way of measuring the CPU capacity.

All the EC2 instances are hyperthreaded.

Step 12 Launch the Cisco Catalyst 8000V instance in the VPC that you created. Use the following properties:

a) Set the **Shutdown** behaviour to **Stop**.
b) Set the **Tenancy** to **Shared**. Choose the Shared option to run a shared hardware instance.

Step 13 Associate the instance with a security group (SSH-ACCESS). The security rules enable you to configure firewall rules to control traffic for your Cisco Catalyst 8000V instance.

Step 14 Associate a private key with the Cisco Catalyst 8000V instance. A key pair is a private key and a public key. You must provide the private key to authenticate and connect to the Cisco Catalyst 8000V instance. The public key is stored on AWS. If required, you can create a new key pair.

Step 15 Click **Launch Instances**.

Step 16 Verify whether the Cisco Catalyst 8000V instance is deployed on AWS.

After successful deployment, the status changes to *2/2/ checks passed*.

Configure subnets

Step 1 Select the Cisco Catalyst 8000V instance.

Step 2 Select **Actions** > **Networking** > **Manage IP Addresses**.

Step 3 Specify the enterprise host address. This IP address is the secondary address of eth1.

Step 4 Click **Yes, Update**.

Configure a tunnel between Cisco Catalyst 8000V on AWS and Cisco Catalyst 8000V on the Enterprise System

The communication between the Cisco Catalyst 8000V instance deployed within the enterprise data center and the Cisco Catalyst 8000V instance deployed within the public cloud is secured by an IP Security (IPsec) tunnel established between them. The LISP-encapsulated traffic is protected with the IPsec tunnel that provides data origin authentication, integrity protection, anti-reply protection, and confidentiality between the public cloud and the enterprise.

Step 1 Configure a Cisco Catalyst 8000V instance on AWS.

```
interface Loopback1
 ip address 33.33.33.33 255.255.255.255
!
interface Tunnel2
 ip address 30.0.0.2 255.255.255.0
 tunnel source GigabitEthernet1
 tunnel mode ipsec ipv4
 tunnel destination 173.39.145.79
 tunnel protection ipsec profile p2p_pf1
!
interface GigabitEthernet2
 ip address 10.10.10.140 255.255.255.0
 negotiation auto
 lisp mobility subnet1 nbr-proxy-reply requests 3
 no mop enabled
 no mop sysid
!
```

Step 2 Configure a second Cisco Catalyst 8000V instance on the enterprise site.

```
interface Loopback1
 ip address 11.11.11.11 255.255.255.255

interface Tunnel2
 ip address 30.0.0.1 255.255.255.0
 tunnel source GigabitEthernet2
 tunnel mode ipsec ipv4
 tunnel destination 52.14.116.161
 tunnel protection ipsec profile p2p_pf1
!
!
interface GigabitEthernet3
 ip address 10.10.10.2 255.255.255.0
 negotiation auto
 lisp mobility subnet1 nbr-proxy-reply requests 3
 no mop enabled
 no mop sysid
!
```

Configure LISP xTR on the Instance Running on AWS

To configure LISP xTR on the Cisco Catalyst 8000V instance running on AWS, follow the configuration steps in the Configuring LISP (Location ID Separation Protocol) section.

Example:

```
router lisp
 locator-set aws
  33.33.33.33 priority 1 weight 100
  exit-locator-set
 !
 service ipv4
  itr map-resolver 11.11.11.11
  itr
  etr map-server 11.11.11.11 key cisco
```

```
  etr
  use-petr 11.11.11.11
  exit-service-ipv4
 !
 instance-id 0
  dynamic-eid subnet1
   database-mapping 10.10.10.0/24 locator-set aws
   map-notify-group 239.0.0.1
   exit-dynamic-eid
  !
  service ipv4
   eid-table default
   exit-service-ipv4
  !
  exit-instance-id
 !
 exit-router-lisp
!
router ospf 11
 network 30.0.0.2 0.0.0.0 area 11
 network 33.33.33.33 0.0.0.0 area 11
!

router lisp
 locator-set dmz
  11.11.11.11 priority 1 weight 100
  exit-locator-set
 !
 service ipv4
  itr map-resolver 11.11.11.11
  etr map-server 11.11.11.11 key cisco
  etr
  proxy-etr
  proxy-itr 11.11.11.11
  map-server
  map-resolver
  exit-service-ipv4
 !
 instance-id 0
  dynamic-eid subnet1
   database-mapping 10.10.10.0/24 locator-set dmz
   map-notify-group 239.0.0.1
   exit-dynamic-eid
  !
  service ipv4
   eid-table default
   exit-service-ipv4
  !
  exit-instance-id
 !
 site DATA_CENTER
  authentication-key cisco
  eid-record 10.10.10.0/24 accept-more-specifics
  exit-site
 !
 exit-router-lisp
!
router ospf 11
 network 11.11.11.11 0.0.0.0 area 11
 network 30.0.0.1 0.0.0.0 area 11
!
```

```
!
!
```

Verify the LISP Layer 2 Traffic between Cisco Catalyst 8000V on AWS and Cisco Catalyst 8000V on the Enterprise System

Perform the following steps to verify the LISP Layer 2 traffic:

Example:

```
Router#show ip lisp database
LISP ETR IPv4 Mapping Database for EID-table default (IID 0), LSBs: 0x1
Entries total 2, no-route 0, inactive 0

10.0.1.1/32, dynamic-eid subnet1, inherited from default locator-set aws
  Locator  Pri/Wgt  Source     State
33.33.33.33    1/100  cfg-addr   site-self, reachable
10.0.1.20/32, dynamic-eid subnet1, inherited from default locator-set aws
  Locator  Pri/Wgt  Source     State
33.33.33.33    1/100  cfg-addr   site-self, reachable
Router#show ip lisp map-cache
LISP IPv4 Mapping Cache for EID-table default (IID 0), 4 entries

0.0.0.0/0, uptime: 00:09:49, expires: never, via static-send-map-request
  Negative cache entry, action: send-map-request
10.0.1.0/24, uptime: 00:09:49, expires: never, via dynamic-EID, send-map-request
  Negative cache entry, action: send-map-request
10.0.1.4/30, uptime: 00:00:55, expires: 00:00:57, via map-reply, forward-native
  Encapsulating to proxy ETR
10.0.1.100/32, uptime: 00:01:34, expires: 23:58:26, via map-reply, complete
  Locator  Uptime    State      Pri/Wgt     Encap-IID
11.11.11.11  00:01:34  up           1/100       -
Router#show lisp dynamic-eid detail
% Command accepted but obsolete, unreleased or unsupported; see documentation.

LISP Dynamic EID Information for VRF "default"

Dynamic-EID name: subnet1
  Database-mapping EID-prefix: 10.0.1.0/24, locator-set aws
  Registering more-specific dynamic-EIDs
  Map-Server(s): none configured, use global Map-Server
  Site-based multicast Map-Notify group: 239.0.0.1
  Number of roaming dynamic-EIDs discovered: 2
  Last dynamic-EID discovered: 10.0.1.20, 00:01:37 ago
    10.0.1.1, GigabitEthernet2, uptime: 00:09:23
      last activity: 00:00:42, discovered by: Packet Reception
    10.0.1.20, GigabitEthernet2, uptime: 00:01:37
      last activity: 00:00:40, discovered by: Packet Reception

Router-DC#show ip lisp
Router-DC#show ip lisp data
Router-DC#show ip lisp database
LISP ETR IPv4 Mapping Database for EID-table default (IID 0), LSBs: 0x1
Entries total 1, no-route 0, inactive 0

10.0.1.100/32, dynamic-eid subnet1, inherited from default locator-set dc
```

```
  Locator  Pri/Wgt  Source     State
11.11.11.11    1/100  cfg-addr   site-self, reachable
Router-DC#show ip lisp
Router-DC#show ip lisp map
Router-DC#show ip lisp map-cache
LISP IPv4 Mapping Cache for EID-table default (IID 0), 2 entries

10.0.1.0/24, uptime: 1d08h, expires: never, via dynamic-EID, send-map-request
  Negative cache entry, action: send-map-request
10.0.1.20/32, uptime: 00:00:35, expires: 23:59:24, via map-reply, complete
  Locator  Uptime    State      Pri/Wgt     Encap-IID
33.33.33.33  00:00:35  up           1/100

Router-DC#show lisp dynamic-eid detail
% Command accepted but obsolete, unreleased or unsupported; see documentation.

LISP Dynamic EID Information for VRF "default"

Dynamic-EID name: subnet1
  Database-mapping EID-prefix: 10.0.1.0/24, locator-set dc
  Registering more-specific dynamic-EIDs
  Map-Server(s): none configured, use global Map-Server
  Site-based multicast Map-Notify group: 239.0.0.1
  Number of roaming dynamic-EIDs discovered: 1
  Last dynamic-EID discovered: 10.0.1.100, 1d08h ago
    10.0.1.100, GigabitEthernet2, uptime: 1d08h
      last activity: 00:00:47, discovered by: Packet Reception

Router-DC#show lisp site
LISP Site Registration Information
* = Some locators are down or unreachable
# = Some registrations are sourced by reliable transport

Site Name      Last      Up     Who Last             Inst     EID Prefix
               Register         Registered           ID
dc             never     no     --                            10.0.1.0/24
               00:08:41  yes#   33.33.33.33                   10.0.1.1/32
               00:01:00  yes#   33.33.33.33                   10.0.1.20/32
               1d08h     yes#   11.11.11.11                   10.0.1.100/32
Router-DC#show ip cef 10.0.1.20
10.0.1.20/32
  nexthop 33.33.33.33 LISP0
Router-DC#
```

Support for PMD Multi-Queue

From Cisco IOS XE 17.7.1, the PMD Multi-Queue functionality is supported on Cisco Catalyst 8000V instances running on AWS. Currently, Cisco Catalyst 8000V allocates only one PMD RX queue and PMD TX queue per interface. With this functionality, Cisco Catalyst 8000V allocates 4 PMD RX queues and 8 PMD TX queues, thereby increasing the performance by increasing the packet processing rate.

From Cisco IOS XE 17.9.1, Cisco Catalyst 8000V has the capacity to allocate 12 PMD TX queues.

Note The IP address pair in the IPsec tunnels are hashed to PMD TXQ. Therefore, the address can cause collision and reduce the performance. To avoid this issue, check whether the traffic is evenly distributed across all the 8 queues for optimal performance by using the **show platform hardware qfp active datapath infrastructure sw-nic** command.

The following is a sample command output of the **show platform hardware qfp active datapath infrastructure sw-nic** command.

```
Router# show platform hardware qfp act datapath infrastructure sw-nic
pmd b19811c0 device Gi1
  RX: pkts 418  bytes 37655 return 0 badlen 0
    pkts/burst 1  cycl/pkt 0  ext_cycl/pkt 0
    Total ring read 91995516, empty 91995113
  TX: pkts 355  bytes 57833
      pri-0: pkts 60  bytes 5590
             pkts/send 1
      pri-1: pkts 32  bytes 2616
             pkts/send 1
      pri-2: pkts 6  bytes 303
             pkts/send 1
      pri-3: pkts 38  bytes 6932
             pkts/send 1
      pri-4: pkts 176  bytes 39279
             pkts/send 1
      pri-5: pkts 25  bytes 1962
             pkts/send 1
      pri-6: pkts 8  bytes 459
             pkts/send 1
      pri-7: pkts 10  bytes 692
             pkts/send 1
  Total: pkts/send 1  cycl/pkt 3160
    send 343  sendnow 0
    forced 343  poll 0  thd_poll 0
    blocked 0  retries 0  mbuf alloc err 0
    TX Queue 0: full 0  current index 0  hiwater 0
    TX Queue 1: full 0  current index 0  hiwater 0
    TX Queue 2: full 0  current index 0  hiwater 0
    TX Queue 3: full 0  current index 0  hiwater 0
    TX Queue 4: full 0  current index 0  hiwater 0
    TX Queue 5: full 0  current index 0  hiwater 0
    TX Queue 6: full 0  current index 0  hiwater 0
    TX Queue 7: full 0  current index 0  hiwater 0
pmd b1717380 device Gi2
  RX: pkts 289216546  bytes 102405925473 return 0 badlen 0
    pkts/burst 7  cycl/pkt 326  ext_cycl/pkt 381
    Total ring read 141222555, empty 103047391
  TX: pkts 757922  bytes 260498122
      pri-0: pkts 94302  bytes 32428428
             pkts/send 1
      pri-1: pkts 95525  bytes 32791822
             pkts/send 1
      pri-2: pkts 93002  bytes 31950500
             pkts/send 1
      pri-3: pkts 96799  bytes 33381108
             pkts/send 1
      pri-4: pkts 90823  bytes 31179044
             pkts/send 1
      pri-5: pkts 97436  bytes 33455916
             pkts/send 1
      pri-6: pkts 93243  bytes 32113540
```

```
              pkts/send 1
          pri-7: pkts 96792  bytes 33197764
              pkts/send 1
    Total: pkts/send 1  cycl/pkt 760
      send 685135  sendnow 3
      forced 685117  poll 0  thd_poll 0
      blocked 0  retries 0  mbuf alloc err 0
      TX Queue 0: full 0  current index 0  hiwater 31
      TX Queue 1: full 0  current index 0  hiwater 31
      TX Queue 2: full 0  current index 0  hiwater 0
      TX Queue 3: full 0  current index 0  hiwater 0
      TX Queue 4: full 0  current index 1  hiwater 31
      TX Queue 5: full 0  current index 0  hiwater 0
      TX Queue 6: full 0  current index 0  hiwater 0
      TX Queue 7: full 0  current index 0  hiwater 0
pmd b14ad540 device Gi3
    RX: pkts 758108  bytes 302121148 return 0 badlen 0
      pkts/burst 1  cycl/pkt 572  ext_cycl/pkt 811
      Total ring read 78867251, empty 78155478
    TX: pkts 756904  bytes 301747138
          pri-0: pkts 9  bytes 540
              pkts/send 1
          pri-1: pkts 200064  bytes 80223776
              pkts/send 1
          pri-3: pkts 244086  bytes 97204792
              pkts/send 1
          pri-4: pkts 3  bytes 822
              pkts/send 1
          pri-5: pkts 250502  bytes 99404344
              pkts/send 1
          pri-7: pkts 62240  bytes 24912864
              pkts/send 1
    Total: pkts/send 1  cycl/pkt 737
      send 705364  sendnow 3
      forced 705355  poll 0  thd_poll 0
      blocked 0  retries 0  mbuf alloc err 0
      TX Queue 0: full 0  current index 0  hiwater 0
      TX Queue 1: full 0  current index 0  hiwater 31
      TX Queue 2: full 0  current index 0  hiwater 0
      TX Queue 3: full 0  current index 0  hiwater 31
      TX Queue 4: full 0  current index 0  hiwater 0
      TX Queue 5: full 0  current index 0  hiwater 0
      TX Queue 6: full 0  current index 0  hiwater 0
      TX Queue 7: full 0  current index 0  hiwater 0
```