

LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: September 4, 2019

S. Barkai
B. Frenandez-Ruiz
O. Serfaty
Nexar Inc.
A. Rodriguez-Natal
F. Maino
Cisco Systems
A. Cabellos-Aparicio
Technical University of Catalonia
February 4 2019

Distributed Geo-Spatial LISP Blackboard for Automotive
draft-barkai-lisp-nexagon-00

Abstract

This document specifies the use of LISP Blackboard for distributed Geo-Spatial Publish/Subscribe automotive applications.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 4, 2018.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	3
3. Deployment Assumptions	3
4. Nexagon Publish Procedure	4
5. Nexagon Subscribe Procedure	5
6. XTR Sharding-Handover tunnel	7
7. Security Considerations	8
8. Acknowledgments	8
9. IANA Considerations	8
10. Normative References	8
Authors' Addresses	8

1. Introduction

(1) The Locator/ID Separation Protocol (LISP) [RFC6830] splits current IP addresses in two different namespaces, Endpoint Identifiers (EIDs) and Routing Locators (RLOCs).

LISP uses a map-and-encap approach that relies on (1) a Mapping System (basically a distributed database) that stores and disseminates EID-RLOC mappings and on (2) LISP tunnel routers (xTRs) that encapsulate and decapsulate data packets based on the content of those mappings.

(2) H3 is a geospatial indexing system using a hexagonal grid that can be (approximately) subdivided into finer and finer hexagonal grids, combining the benefits of a hexagonal grid with hierarchical subdivisions. H3 supports sixteen resolutions. Each finer resolution has cells with one seventh the area of the coarser resolution. Hexagons cannot be perfectly subdivided into seven hexagons, so the finer cells are only approximately contained within a parent cell. Each cell is identified by a 64bit int.

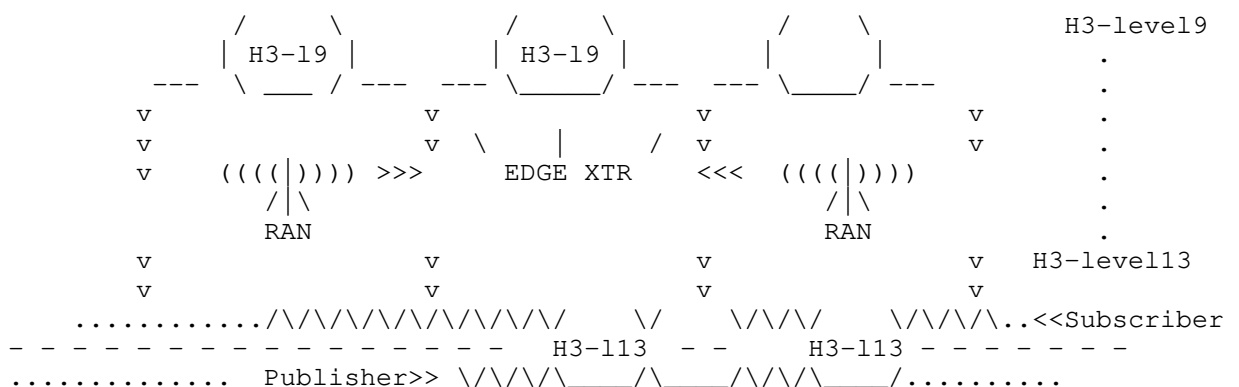
(3) The Berkeley Deep Drive (BDD) Industry Consortium investigates state-of-the-art technologies in computer vision and machine learning for automotive applications. BDD based taxonomy of published automotive scene classification.

These standards are combined to create an in-network key-value blackboard - reflecting the state of each 1sqm hexagon tile of road. The lisp network maps traffic from vehicle endpoint IP identifiers (EID) to the routing location (RLOC) of h3 hexagon identifier (HID).

Th lisp network blackboard bridges the time-space gap between vision & sensory (publishers) - and - driving apps/smart-infrastructure (subscribers). Drivers (EID) communicate with blackboard tiles (HID), EID<=> RLOC <=> HID, small tiles to publish, large tiles to subscribe to regional information.

One of of the key use-cases is providing drivers with 20-30 seconds preemptive heads-up on potential hazards and obstacles; across traffic, around the block, beyond turns and curvatures, in a nutshell beyond sensory line-of-site.

- (1) LISP blackboard keys are 64bit H3 IDs referring to ~1sqm H3 level 13
- (2) LISP blackboard values are 64bit compiled-states of each H3 road-tile
- (3) LISP blackboard pub-sub regions are at H3 level-9 containing 113 tiles
- (4) LISP Blackboard is sharded to scale state-updates and edge propagation
- (5) Edge LISP XTRs use the H3 IDs to map publish-subscribe to right shard
- (6) Edge XTRs are also used to replicate bulk state updates to clients
- (7) Bulk updates Multicast-replication can use native access multicast



2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Deployment Assumptions

The specification described in this document makes the following deployment assumptions:

- (1) A unique 64-bit H3 Hex-Tile identifier is associated with each lang-lat
- (2) Clients (Publisher/Subscriber) and network (Blackboard) share this index
- (3) A 64-bit automotive BDD state value is associated with each hexagon tile
- (4) Hexagon state is combined by 16 fields of 4-bit (nibble) up-to 16 enums

```
| -0- | -1- | -2- | -3- | -4- | -5- | -6- | -7- | -8- | -9- | -A- | -B- | -C- | -D- | -E- | -F- |
01230123012301230123012301230123012301230123012301230123012301230123012301230123
```

- (5) The following fields describe state information for a given tile

Field 0x describes the "freshness" of the state eg last published {

```
0x: less than 10Sec
1x: less than 20Sec
2x: less than 40Sec
3x: less than 1min
4x: less than 2min
5x: less than 5min
6x: less than 15min
7x: less than 30min
9x: less than 1hour
Ax: less than 2hours
Bx: less than 8hours
Cx: less than 24hours
Dx: less than 1week
Ex: less than 1month
Fx: more than 1month
```

}

field 1x: persistent weather or structural {

```
0x - null
1x - pothole
2x - speed-bump
3x - icy
4x - flooded
5x - snow-cover
6x - snow-deep
7x - construction cone
8x - curve
```

}

field 2x: transient or moving obstruction {

```
0x - null
1x - pedestrian
2x - bike
3x - stopped car / truck
4x - moving car / truck
5x - first responder vehicle
6x - sudden slowdown
7x - oversized-vehicle
```

}

field 3x: traffic-light timer countdown {

```
0x - green now
1x - 1 seconds to green
2x - 2 seconds to green
3x - 3 seconds to green
4x - 4 seconds to green
5x - 5 seconds to green
6x - 6 seconds to green
7x - 7 seconds to green
8x - 8 seconds to green
```

```

    9x - 9 seconds to green
    Ax - 10 seconds or less
    Bx - 20 seconds or less
    Cx - 30 seconds or less
    Dx - 40 seconds or less
    Ex - 50 seconds or less
    Fx - minute or more left
}

field 4x: impacted tile from neighboring {
    0x - not impacted
    1x - light yellow
    2x - yellow
    3x - light orange
    4x - orange
    5x - light red
    6x - red
    7x - light blue
    8x - blue
}

field 5x: incidents {
    0x - clear
    1x - light collision (fender bender)
    2x - hard collision
    3x - collision with casualty
    4x - recent collision residues
    5x - hard break
    6x - sharp cornering
}

field 6x - compiled tile safety rating {

}

field 7x - reserved
field 8x - reserved
field 9x - reserved
field Ax - reserved
field Bx - reserved
field Cx - reserved
field Dx - reserved
field Ex - reserved
field Fx - reserved

```

(7) Publish packet contains 1 key-value tuple:

-0-	-1-	-2-	-3-	-4-	-5-	-6-	-7-	-8-	-9-	-A-	-B-	-C-	-D-	-E-	-F-
H3 Hexagon ID Key															
-0-	-1-	-2-	-3-	-4-	-5-	-6-	-7-	-8-	-9-	-A-	-B-	-C-	-D-	-E-	-F-
H3 Hexagon State-Value															

- (8) Any number of fields published in a state can be set to a value
- (9) If a field is not being addressed by then it should be set to 0x-null
- (10) Subscribe packets are the same as publish with the entire state set null

4. Nexagon Publish-Procedure

- (1) Publisher observation
- (2) Snap to hex accuracy bar
- (3) Compiling a Publish Packet
- (4) Publish Packet Source IP
- (5) Publish Packet Destination IP

5. Nexagon Subscribe Procedure

- (1) Subscribe to zone hierarchy
- (2) Subscribe Packet
- (3) Zone state update packet of upto 100 hexagon tiles

1	-0- -1- -2- -3- -4- -5- -6- -7- -8- -9- -A- -B- -C- -D- -E- -F-
	H3 Hexagon ID Key
	-0- -1- -2- -3- -4- -5- -6- -7- -8- -9- -A- -B- -C- -D- -E- -F-
	H3 Hexagon State-Value

.	.
.	.
.	.
100	-0- -1- -2- -3- -4- -5- -6- -7- -8- -9- -A- -B- -C- -D- -E- -F-
	H3 Hexagon ID Key
	-0- -1- -2- -3- -4- -5- -6- -7- -8- -9- -A- -B- -C- -D- -E- -F-
	H3 Hexagon State-Value

6. XTR Sharding and Handover to blackboard tunnels

- (1) Map-Resolve hexagon ID to shard location
- (2) Multicast replication to subscribed EIDs

7. Security Considerations

The way to provide a security association between the ITRs and the Map-Servers must be evaluated according to the size of the deployment. For small deployments, it is possible to have a shared key (or set of keys) between the ITRs and the Map-Servers. For larger and Internet-scale deployments, scalability is a concern and further study is needed.

8. Acknowledgments

This work is partly funded by the ANR LISP-Lab project #ANR-13-INFR-009 (<https://lisplab.lip6.fr>).

9. IANA Considerations

This document makes no request to IANA.

10. Normative References

- [I-D.ietf-lisp-rfc6833bis]
Fuller, V., Farinacci, D., and A. Cabellos-Aparicio,
"Locator/ID Separation Protocol (LISP) Control-Plane",
draft-ietf-lisp-rfc6833bis-07 (work in progress), December
2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The
Locator/ID Separation Protocol (LISP)", RFC 6830,
DOI 10.17487/RFC6830, January 2013,
<<https://www.rfc-editor.org/info/rfc6830>>.

Authors' Addresses

Sharon Barkai
Nexar

CA
USA

Email: sharon.barkai@getnexar.com

Bruno Fernandez-Ruiz
Nexar
London
UK

Email: b@getnexar.com

Ohad Serfaty
Nexar
Israel

Email: ohad@getnexar.com

Alberto Rodriguez-Natal
Cisco Systems
170 Tasman Drive
San Jose, CA
USA

Email: natal@cisco.com

Fabio Maino
Cisco Systems
170 Tasman Drive
San Jose, CA
USA

Email: fmaino@cisco.com

Albert Cabellos-Aparicio
Technical University of Catalonia
Barcelona
Spain

Email: acabello@ac.upc.edu

LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: March 10, 2020

S. Barkai
B. Fernandez-Ruiz
S. ZionB
R. Tamir
Nexar Inc.
A. Rodriguez-Natal
F. Maino
Cisco Systems
A. Cabellos-Aparicio
J. Paillissé Vilanova
Technical University of Catalonia
D. Farinacci
lispers.net
October 10, 2019

Network-Hexagons: H3-LISP Based Mobility Network
draft-barkai-lisp-nexagon-11

Abstract

This document specifies combined use of H3 and LISP for mobility-networks:
- Enabling real-time tile by tile indexed annotation of public roads
- For sharing: hazards, blockages, conditions, maintenance, furniture..
- Between MobilityClients producing-consuming road geo-state information
- Using addressable grid of channels of physical world state representation

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 4, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Requirements Language	3
3. Definition of Terms	3
4. Deployment Assumptions	4

5. Mobility Clients-Network-Services	4
6. Mobility Unicast-Multicast	5
7. Security Considerations	6
8. Acknowledgments	6
9. IANA Considerations	6
10. Normative References	8
Authors' Addresses	9

1. Introduction

(1) The Locator/ID Separation Protocol (LISP) [RFC6830] splits current IP addresses in two different namespaces, Endpoint Identifiers (EIDs) and Routing Locators (RLOCs). LISP uses a map-and-encap approach that relies on (1) a Mapping System (distributed database) that stores and disseminates EID-RLOC mappings and on (2) LISP tunnel routers (xTRs) that encapsulate and decapsulate data packets based on the content of those mappings.

(2) H3 is a geospatial indexing system using a hexagonal grid that can be (approximately) subdivided into finer and finer hexagonal grids, combining the benefits of a hexagonal grid with hierarchical subdivisions. H3 supports sixteen resolutions. Each finer resolution has cells with one seventh the area of the coarser resolution. Hexagons cannot be perfectly subdivided into seven hexagons, so the finer cells are only approximately contained within a parent cell. Each cell is identified by a 64bit HID.

(3) The Berkeley Deep Drive (BDD) Industry Consortium investigates state-of-the-art technologies in computer vision and machine learning for automotive applications, and, for taxonomy of published automotive scene classification.

These standards are combined to create in-network-state which reflects the condition of each hexagon tile (~1sqm) in every road. The lisp network maps & encapsulates traffic between MobilityClients endpoint-identifiers (EID), and, addressable (HID=>EID) tile-states. States are aggregated byH3Service EIDs.

The H3-LISP mobility network bridges timing-location gaps between the production and consumption of information by MobilityClients:

- o vision, sensory, LIADR, AI applications - information producers
- o driving-apps, smart-infrastructure, command & control - who consume it

This is achieved by putting the physical world on a shared addressable geo-state grid at the edge, a low-latency production-consumption indirection. Tile by tile based geo-state mobility-network solves key issues in todays' vehicle to vehicle networking, where observed hazards are expected to be relayed or "hot-potato-tossed" (v2v without clear-reliable convergence i.e. given a situation observable by some of traffic, it is unclear if the rest of the relevant traffic will receive consistent, conflicting, multiple, or no indication what so ever - using peer-to-peer propagation.

For example, when a vehicle experiences a sudden highway slow-down, "sees" many brake-lights or "feels" accelerometer, there is no clear way for it to share this annotation with vehicles 20-30sec away for preventing potential pile-up. Or, when a vehicle crosses an intersection, observing opposite-lane obstruction - construction, double-park, commercial-loading / un-loading, garbage truck, or stopped school-bus - there is no clear way for it to alert vehicles turning in to that situation as it drives away.

Geo-state indirection also helps solve communicating advanced machine-vision and radar annotations. These are constantly evolving technologies, however, communicating the road enumerations they produce using peer-to-peer protocols poses a significant interoperability challenge - testing each new annotation by any sensor / OEM vendor and any other OEM and driving application vendor.

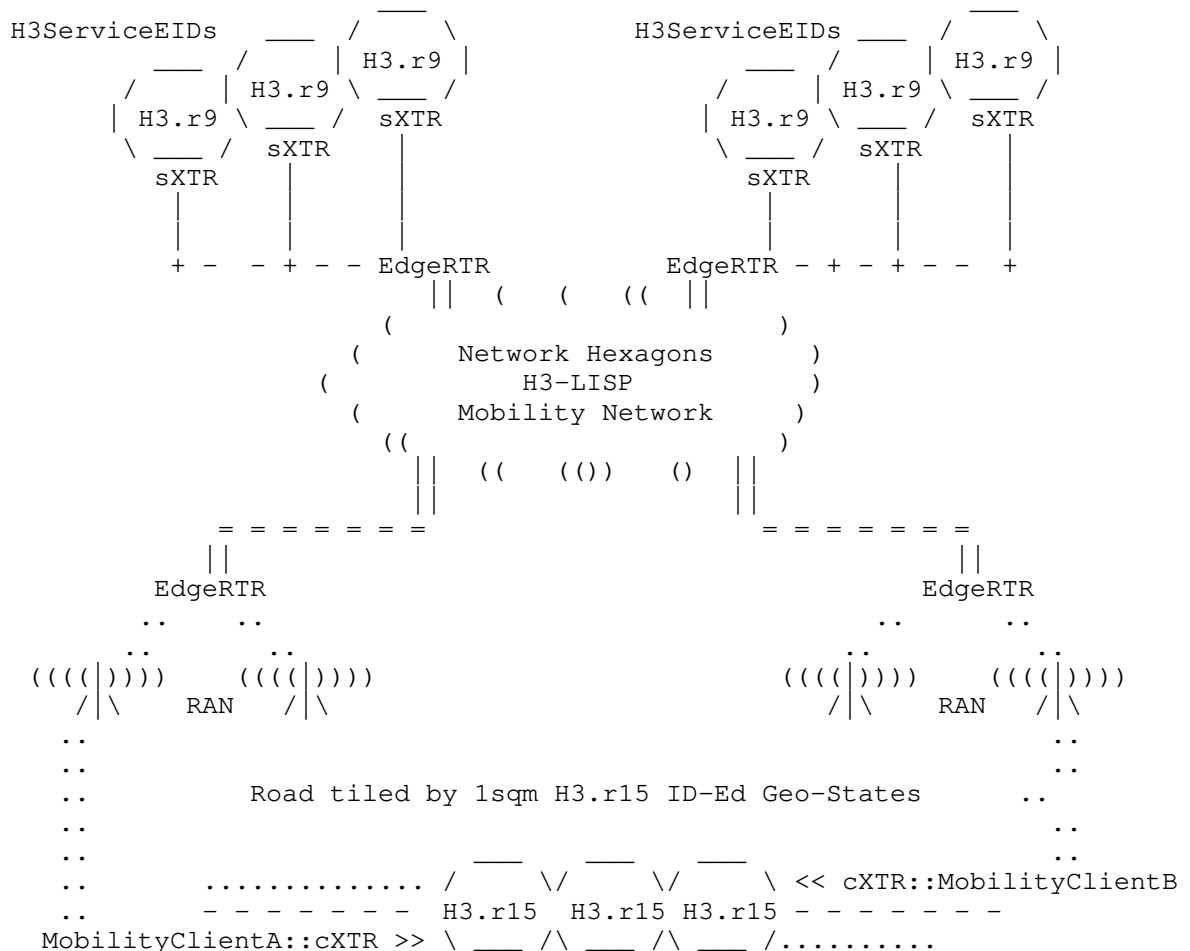
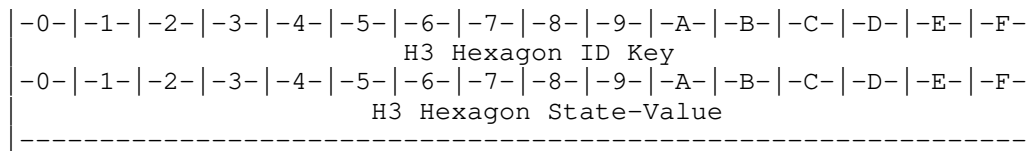
These peer-to-peer limitations are inherit yet unnecessary, as in most road situations vehicles are not really proper peers. They just happen to be in the same place at the same time. The H3-LISP mobility network solves limitations of direct vehicle to vehicle communication because it anchors per each geo-location: timing, security, privacy, interoperability. Anchoring is by

MobilityClients communicating through in-network geo-states. Addressable tiles are aggregated and maintained by LISP H3ServiceEIDs.

An important set of use-cases for state propagation of information to MobilityClients is to provide drivers heads-up alerts on hazards and obstacles beyond line of sight of both the drivers and in-car sensors: over traffic, around blocks, far-side-junction, beyond turns, and surface-curvatures. This highlights the importance of networks in providing road-safety.

To summarize the H3-LISP solution outline:

- (1) MicroPartition: 64bit indexed geo-spatial H3.r15 road-tiles
- (2) EnumState: 64bit state values compile tile condition representation
- (3) Aggregation: H3.r9 H3ServiceEID group individual H3.r15 road-tiles
- (4) Channels: H3ServiceEIDs function as multicast state update channels
- (5) Scale: H3ServiceEIDs distributed for in-network for latency-throughput
- (6) Mapped Overlay: tunneled-network routes the mobility-network traffic
- (7) Signal-free: tunneled overlay is used to map-register for mcast channels
- (8) Aggregation: tunnels used between MobilityClients/H3ServiceEIDs <> edge
- (9) Access: ClientXTRs/ServerXTRs tunnel traffic to-from the LISP EdgeRTRs
- (10) Control: EdgeRTRs register-resolve H3ServiceEIDs and mcast subscription



- MobilityClientA has seen MobilityClientB (20-30 sec) future, and, vice versa

- Clients share information using addressable shared-state routed by LISP Edge
- ClientXTR (cXTR): tunnel encapsulation through access network to LISP Edge
- ServerXTR (sXTR): tunnel encapsulation through cloud network to LISP Edge
- The H3-LISP Mobility overlay starts in the cXTR and terminates in the sXTR
- The updates are routed to the appropriate tile geo-state by the LISP network
- EdgeRTRs perform multicast replication to edges and then native or to cXTRs
- Clients receive tile-by-tile geo-state updates via the multicast channels

Each H3.r9 hexagon is an EID Service with corresponding H3 hexagon ID. Bound to that service is a LISP xTR, called a ServerXTR, resident to deliver encapsulated packets to and from the H3ServiceEID and LISP Edge. EdgeRTRs are used to re-tunnel packets from MobilityClients to H3ServiceEIDs. Each H3ServiceEID is also a source multicast address for updating MobilityClients on the state of the H3.r15 tiles aggregated-represented by the H3ServiceEID.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Definition of Terms

H3ServiceEID: Is an addressable aggregation of H3.r15 state-tiles. It is a designated source for physical world reported annotations, and an (s,g) source of multicast public-safety update channels. H3ServiceEID is itself an H3 hexagon, large enough to provide geo-spatial conditions context, but not too large as to over-burden (battery powered, cellular connected) subscribers with too much information. For Mobility Network it is H3.r9. It has a light-weight LISP protocol stack to tunnel packets aka ServerXTR. The EID is an IPv6 EID that contains the H3 64-bit address numbering scheme. See IANA consideration for details.

ServerXTR: Is a light-weight LISP protocol stack implementation that co-exists with H3ServiceEID process. When the server roams, the xTR roams with it. The ServerXTR encapsulates and decapsulates packets to/from EdgeRTRs.

MobilityClient: Is a roaming application that may be resident as part of an automobile, as part of a navigation application, part of municipal, state, or federal government command and control application, or part of live street view consumer type of application. It has a light-weight LISP protocol stack to tunnel packets aka ClientXTR.

MobilityClient EID: Is the IPv6 EID used by the Mobility Client applications to source packets. The destination of such packets are only H3ServiceEIDs. The EID format is opaque and is assigned as part of the MobilityClient network-as-a-service (NaaS) authorization.

ClientXTR: Is the light-weight LISP protocol stack implementation that is co-located with the Mobility Client application. It encapsulates packets sourced by applications to EdgeRTRs and decapsulates packets from EdgeRTRs.

EdgeRTR: Is the core scale and structure of the LISP mobility network. EdgeRTRs proxy H3ServiceEIDs and MobilityClient H3ServiceEID channel registration. EdgeRTRs aggregate MobilityClients and H3Services using tunnels to facilitate hosting-providers and mobile-hosting flexibility - for accessing the nexagon mobility network. EdgeRTRs decapsulate packets from ClientXTRs and ServerXTRs and re-encapsulates packets to the clients and servers tunnels. EdgeRTRs glean H3ServiceEIDs and glean MobilityClient EIDs when it decapsulates packets. EdgeRTRs store H3ServiceEIDs and their own RLOC of where the H3ServiceEID is currently reachable from in the map-cache. These mappings are registered to the LISP mapping system so other EdgeRTRs know where to encapsulate for such EIDs. EdgeRTRs do not register MobilityClients' EIDs at the mapping service as

these are temporary-renewed while using the mobility network. Enterprises may provide their own client facing EdgeRTRs to mask their clients geo-whereabouts while using the mobility network.

4. Deployment Assumptions

The specification described in this document makes the following deployment assumptions:

- (1) Unique 64-bit HID is associated with each H3 geo-spatial tile
- (2) MobilityClients and H3ServiceEIDs share this well known index
- (3) 64-bit BDD state value is associated with each H3-indexed tile
- (4) Tile state is compiled 16 fields of 4-bits, or max 16 enums

```
| -0- | -1- | -2- | -3- | -4- | -5- | -6- | -7- | -8- | -9- | -A- | -B- | -C- | -D- | -E- | -F- |
0123012301230123012301230123012301230123012301230123012301230123
```

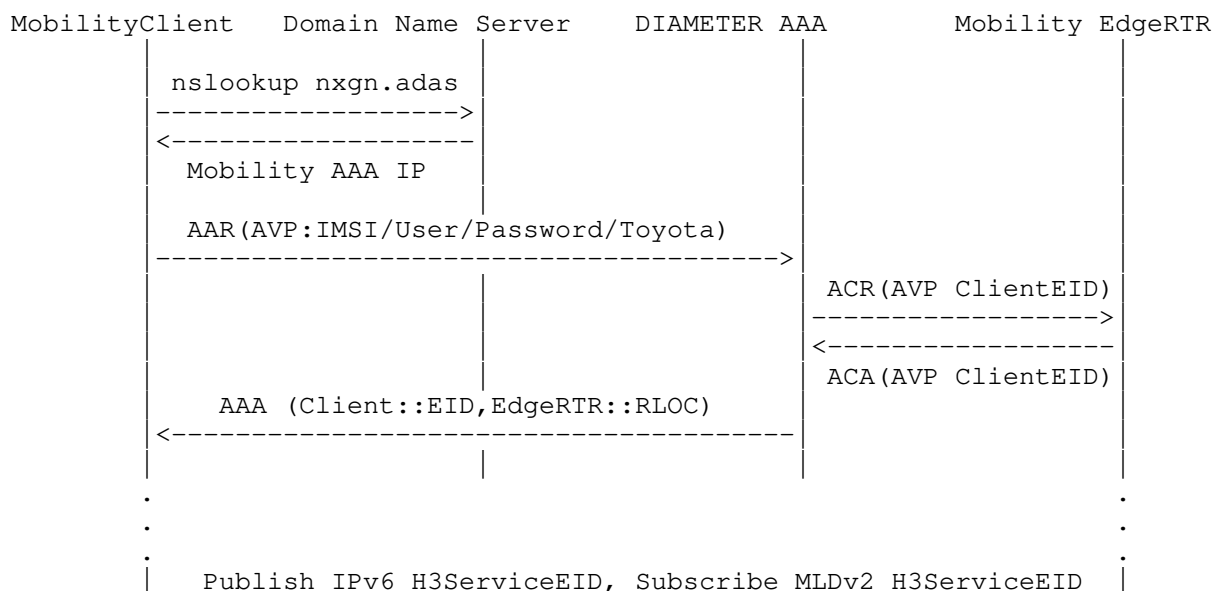
Subscription of MobilityClients to the mobility network is temporary-renewed while on the move and is not intended as means of basic connectivity. This is why MobilityClients use DNS/AAA to obtain temporary EIDs and EdgeRTRs and why they use (LISP) data-plane tunnels to communicate using their temporary EIDs with the dynamically assigned EdgeRTRs.

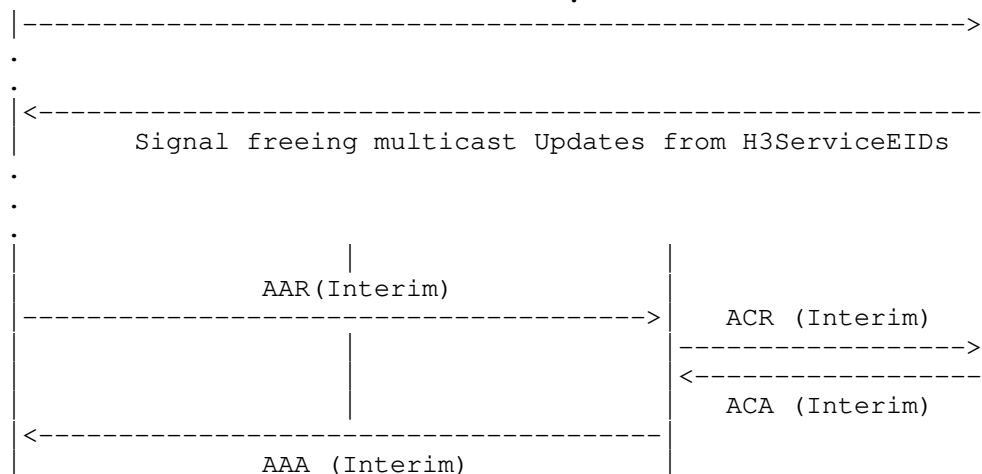
MobilityClient are otherwise unaware of the LISP network mechanism or mapping system and simply regard the data-plane tunnels application specific virtual private network (VPN) that supports IPv6 EID addressable geo-state for publis
h
(Ucast), Subscribe (Mcast) H3Services.

In order to get access to the MobilityVPN MobilityClients first authenticate with the MobilityVPN AAA Server. DIAMETER based AAA is typically done at the provider-edge PE by edge gateways. However the typical case involves handful of customer-premise equipment (CPE/UE) types physically connected by wireline, or, by wireless spectrum to a specific service-provider. The Mobility VPN overlays potentially a number of wireless network providers and cloud-edge providers, and it involves dozens of Car-OEM, Driving-Applications, Smart-infrastructure vendors. It is therefore required to first go through AAA in-order to get both a MobilityClientEID and EdgeRTR gateway RLOC opened.

ClientXTR performs the following steps in-order to use the mobility network:

- 1) obtain the address of the mobility network AAA server using DNS
- 2) obtain MobilityClientEID and EdgeRTR(s) from AAA server using DIAMETER
- 3) renew authorization from AAA while using the mobility network T1 minutes





Using this network-login / re-login method we ensure that:

- the MobilityClientEIDs serve as credentials with the specific EdgeRTRs
- EdgeRTRs are not tightly coupled to H3.r9 areas for privacy/load-balance
- Mobility Clients do not need to update EdgeRTRs while roaming in a metro

The same EdgeRTR may serve several H3.r9 areas for smooth ride continuity, and, several EdgeRTRs may load balance a H3.r9 area with high density of originating MobilityClient rides. When a MobilityClient ClientXTR is homed to EdgeRTR it is able to communicate with H3ServiceEIDs.

5. Mobility Clients-Network-Services

The mobility network functions as a standard LISP VPN overlay.

The overlay delivers unicast and multicast packets across:

- multiple access-network-providers / radio-access-technologies.
- multiple cloud-edge hosting providers, public, private, hybrid.

We use data-plane XTRs in the stack of each mobility client and server. ClientXTRs and ServerXTRs are homed to one or more EdgeRTRs at the LISP edge. This structure allows for MobilityClients to "show-up" at any time, behind any network-provider in a given mobility network administrative domain (metro), and for any H3ServiceEID to be instantiated, moved, or failed-over to - any rack in any cloud-provider. The LISP overlay enables these roaming mobility network elements to communicate un-interrupted. This quality is insured by the LISP RFCs. The determinism of identities for MobilityClients to always refer to the correct H3ServiceEID is insured by H3 geospatial HIDs.

There are two options for how we associate ClientXTRs with LISP EdgeRTRs:

I. Semi-random load-balancing by DNS/AAA

In this option we assume that in a given metro edge a pool of EdgeRTRs can distribute the Mobility Clients load randomly between them and that EdgeRTRs are topologically more or less equivalent. Each RTR uses LISP to tunnel traffic to and from other EdgeRTRs for MobilityClient with H3Service exchanges

MobilityClients can (multi) home to EdgeRTRsRTRs throughout while moving.

II. Topological by any-cast

In this option we align an EdgeRTR with topological aggregation like in the Evolved Packet Core (EPC) solution. Mobility Clients currently roaming in an area home to that RTR and so is the H3 Server. There is only one hop across the edge overlay between clients and servers and mcast replication is more focused, but clients need to keep re-homing as they move.

To summarize the H3LISP mobility network layout:

- (1) Mobility-Clients traffic is tunneled via data-plane ClientXTRs
ClientXTRs are (multi) homed to EdgeRTR(s)
- (2) H3ServiceEID traffic is tunneled via data-plane ServerXTR
ServerXTRs are (multi) homed to EdgeRTR(s)
- (3) EdgeRTRs use mapping service to resolve Ucast HIDs to RTR RLOCs
EdgeRTRs also register to (Source, Group) H3ServiceEID multicasts

```

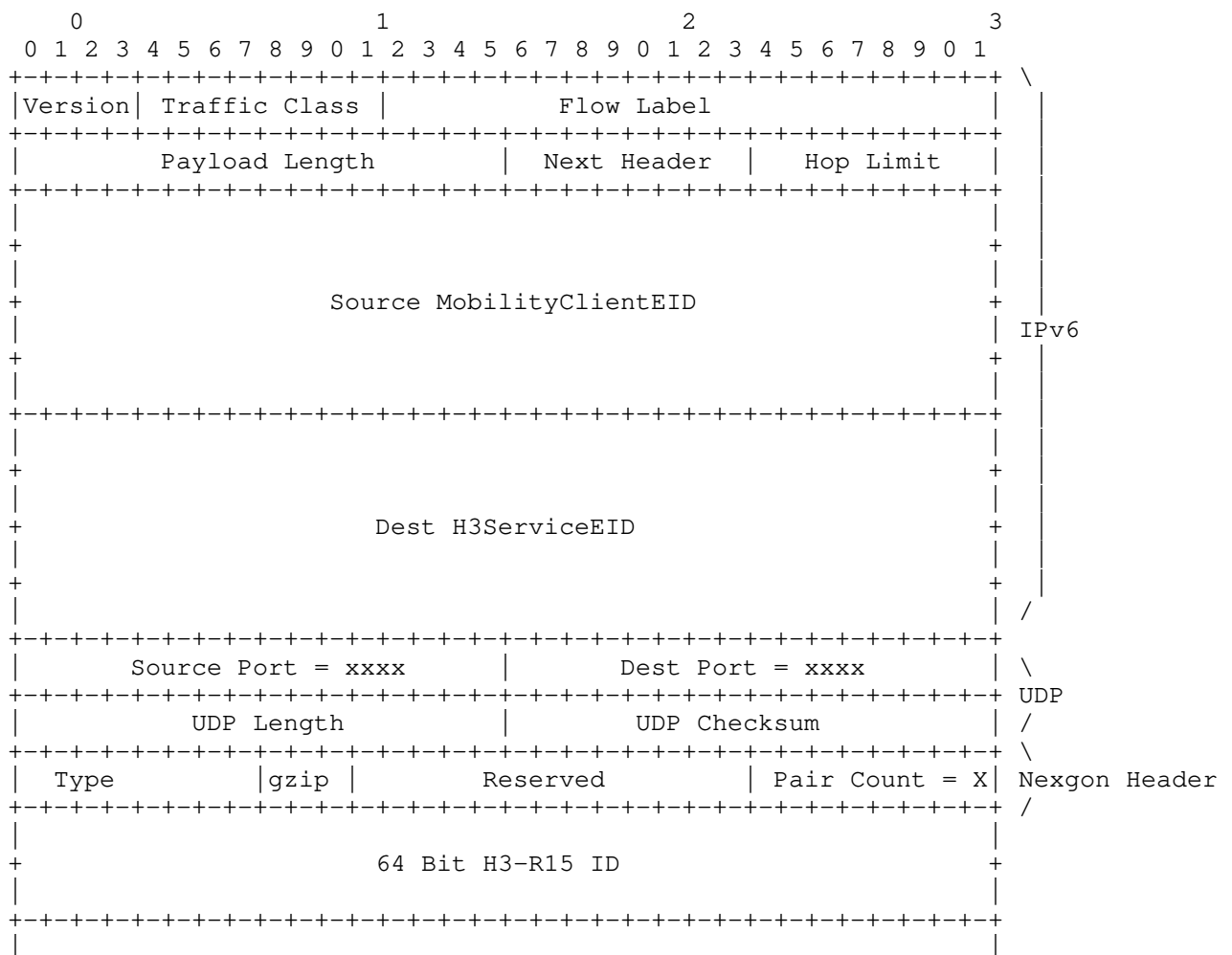
MobilityClients <> ClientXTR <Access Provider > EdgeRTR  v
                                                    v
v      << Map-Assisted Mobility-Network Overlay <<      v
v
>> EdgeRTR <Cloud Provider> ServerXTR <> H3ServiceEID

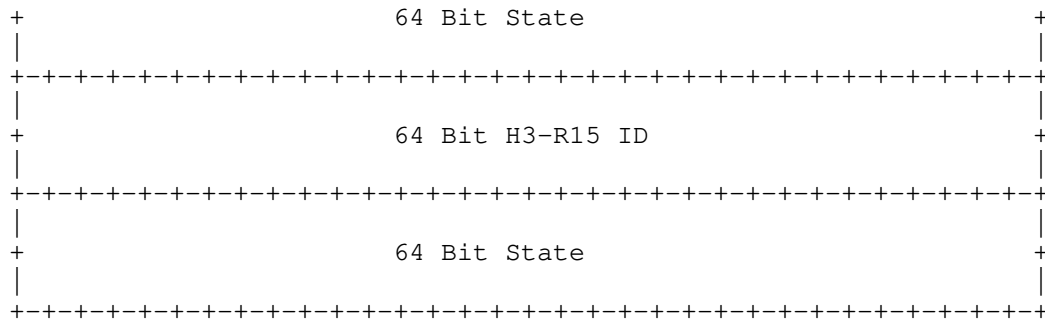
```

6. Mobility Unicast and Multicast

Which ever way a ClientXTR is homed to an Edge RTR an authenticated MobilityClient EID can send: [64bitH3.15ID :: 64bitState] annotation to the H3.r9 H3ServiceEID. The H3.r9 IP HID can be calculated by clients algorithmically form the H3.15 localized snapped-to-tile annotation.

The ClientXTR encapsulates MobilityClient EID and H3ServiceEID in a packet sourced from the ClientXTR, destined to the EdgeRTR RLOC IP, Lisp port. EdgeRTRs then re-encapsulate annotation packets either to remote EdgeRTR (option1) or to homed H3ServiceEID ServerXTR (option2). The remote EdgeRTR aggregating H3ServiceEIDs re-encapsulates MobilityClient EID to ServerXTR and from there to the H3ServiceEID.





To Summarize Unicast:

- (1) MobilityClients can send annotation state localized an H3.r15 tile
These annotations are sent to an H3.r9 mobility H3ServiceEIDs
- (2) MobilityClient EID and H3ServiceEID HID are encapsulated:
XTR <> RTR <> RTR <> XTR
* RTRs can map-resolve re-tunnel HIDs
- (3) RTRs re-encapsulate original source-dest to ServerXTRs
ServerXTRs decapsulate packets to H3ServiceEID

Each H3.r9 Server is used by clients to update H3.r15 tile state is also an IP Multicast channel Source used to update subscribers on the aggregate state of the H3.r15 tiles in the H3.r9 Server.

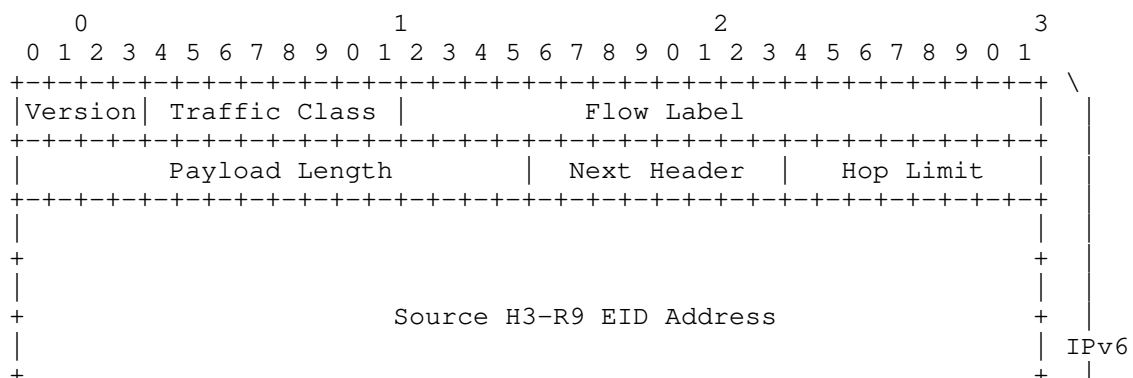
We use rfc8378 signal free multicast to implement mcast channels in the overlay. The mobility network has many channels and relatively few subscribers per each. MobilityClients driving through or subscribing to a H3.r9 area can explicitly issue an rfc4604 MLDv2 in-order to subscribe, or, may be subscribed implicitly by the EdgeRTR glean to ucast HID dest.

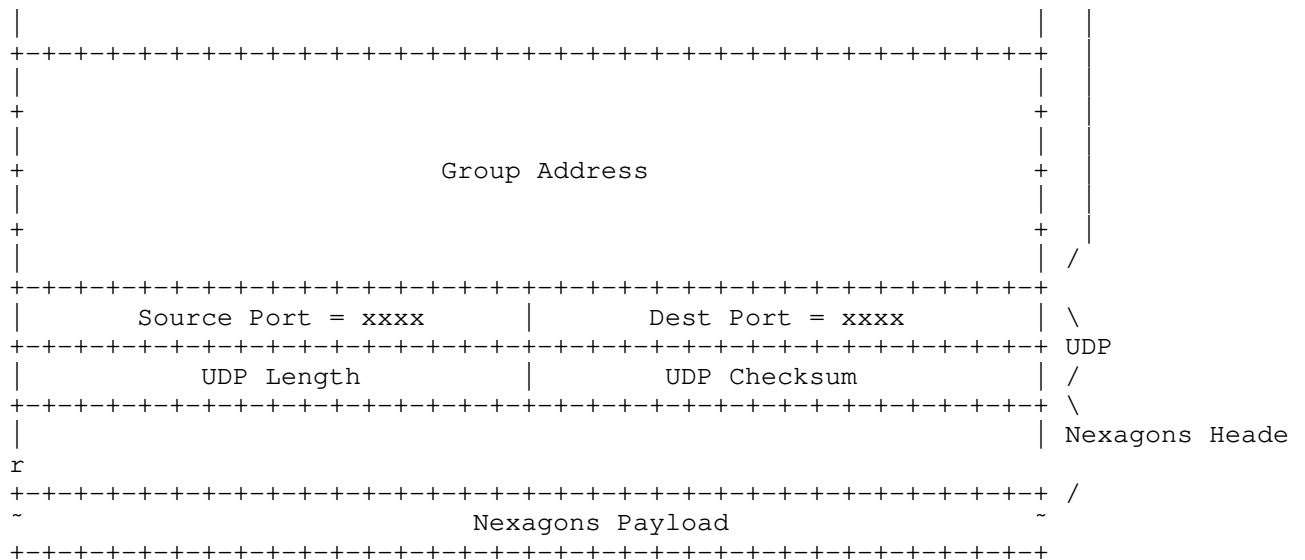
The advantage of explicit client MLDv2 registration trigger to rfc8378 is that the clients manage their own mobility mcast hand-over according to their location-direction moment vectors, and that it allows for otherwise silent, or

, non annotating clients. The advantage of EdgeRTR implicit registration is less signaling required.

MLDv2 signaling messages are encapsulated between the ClientXTR and the LISP EdgeRTR, therefore there is no requirement for the underlying network to support native multicast. If native access multicast is supported (for example native 5G multicast), then MobilityClient registration to H3ServiceEID safety channels may be integrated to it, in which case the evolved-packet-core (EPC) element supporting it (eNB) will use this standard to register with the appropriate H3.r9 channels in its area.

Multicast update packets are of the following structure:

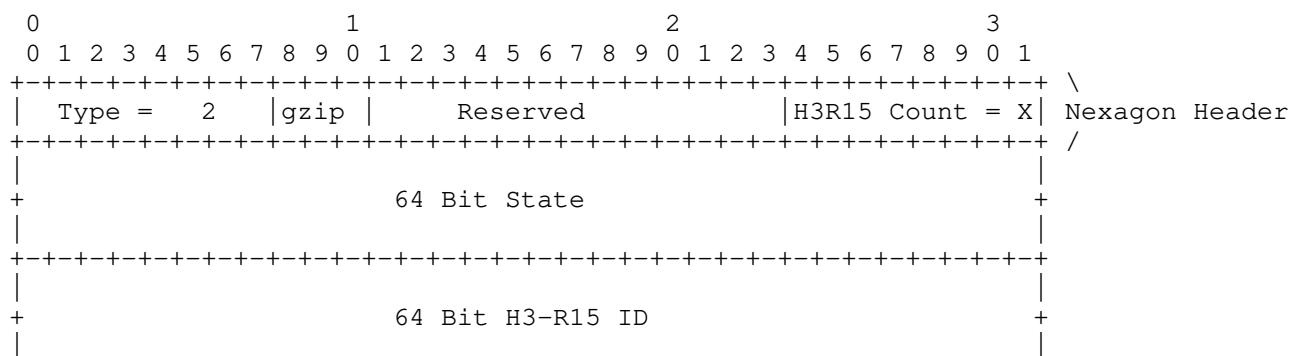
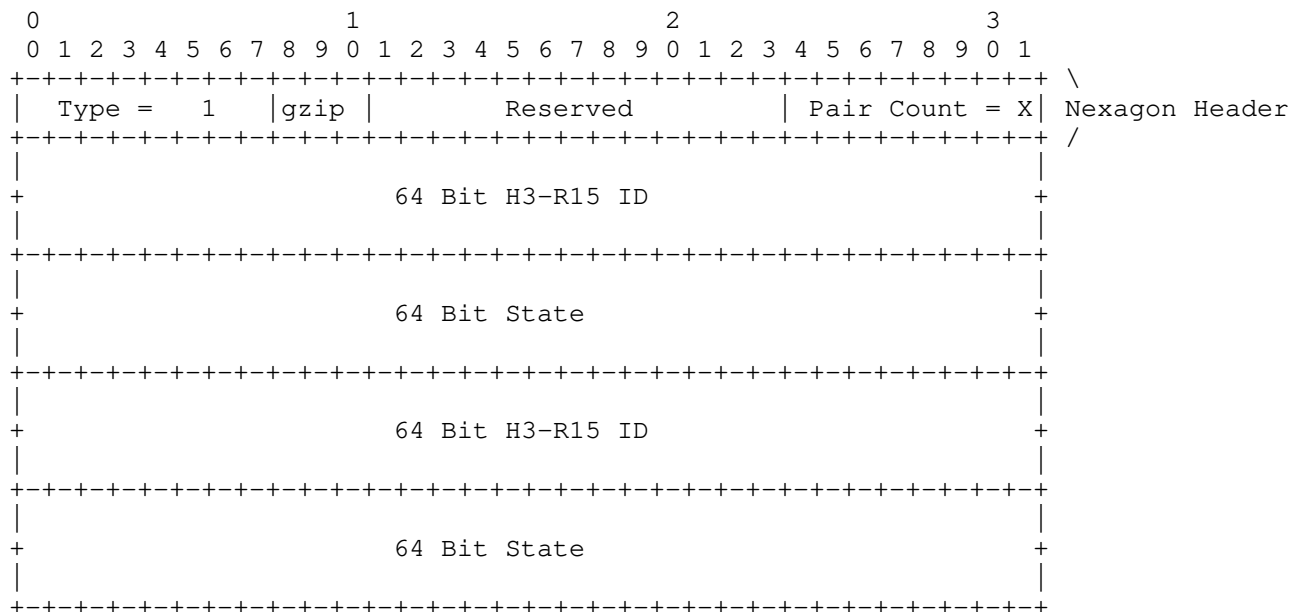


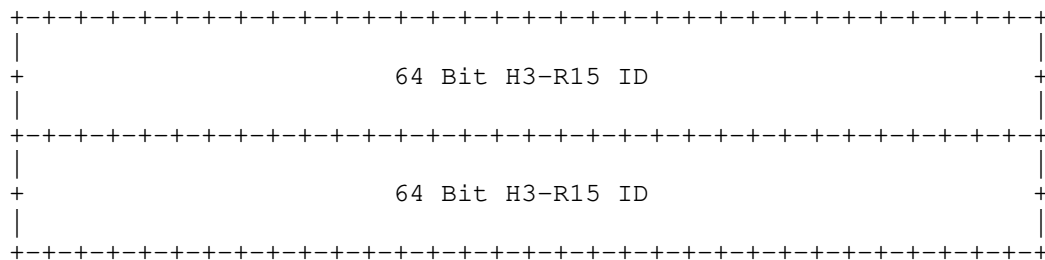


Outer headers = 40 (IPv6) + 8 (UDP) + 8 (LISP) = 56
 Inner headers = 40 (IPv6) + 8 (UDP) + 4 (Nexagon Header) = 52

1500 (MTU) - 56 - 52 = 1392 bytes of effective payload

Type 1:key-value, key-value.. 1392 / (8 + 8) = 87 pairs
 Type 2:value, key,key,key.. (1392 - 8) / 8 = 173 H3-R15 IDs





• The remote EdgeRTRs homing MobilityClients in-turn replicate the packet to the MobilityClients registered with them.

We expect an average of 600 H3.r15 tiles of the full 7^6 (~100K) possible in H3.r9 to be part of any road. The H3.r9 server can transmit the status of all 600 or just those with meaningful state based on update SLA and policy.

To Summarize:

- (1) H3LISP Clients tune to H3.r9 mobility updates using rfc8378
H3LISP Client issue MLDv2 registration to H3.r9 HIDs
ClientXTRs encapsulate MLDv2 to EdgeRTRs who register (s,g)
- (2) ServerXTRs encapsulate updates to EdgeRTRs who map-resolve (s,g) RLOCs
EdgeRTRs replicate mobility update and tunnel to registered EdgeRTRs
Remote EdgeRTRs replicate updates to registered ClientXTRs

7. Security Considerations

The nexagon layer3 v2v/v2i/c&c network is inherently more secure and private than alternatives because of the indirection. No car or infrastructure element ever communicates directly with MobilityClients. All information is conveyed using shared / addressable geo-state. MobilityClients are supposed to receive information only from the network as a trusted broker without indication as to the origin of the information. This is an important step towards better privacy, security, extendability, and interoperability.

In order to be able to use the nexagon mobility network for a given period, the mobility clients go through a DNS/AAA stage by which they obtain their clientED identifiers-credentials and the RLOCs of EdgeRTRs they may use as gateways to the network. This MobilityClient <> EdgeRTR is the most sensitive interface in the network as far as privacy-security.

The traffic on the MobilityClient<>EdgeRTR interface is tunneled and its UDP content may be encrypted, still, the EdgeRTR will know based on the LISP headers alone the MobilityClient RLOC and H3-R9 (~0.1sqkm) geo-spatial area a given client publishes in or subscribes to.

For this reason we envision the ability of enterprise or groups of users to "bring their own" EdgeRTRs. BYO-RTR masks individual clients' IP-RLOC to H3-R9 association and is pre-provisioned to be able to use the mapping system and be on a white-list of EdgeRTRs aggregating H3ServiceEIDs.

Beyond this sensitive hop, the mapping system does not hold MobilityClientEIDs and remote EdgeRTRs are only aware of MobilityClient ephemeral EIDs not their actual IP RLOC or any other mobile-device identifiers. EdgeRTRs register in the mapping (s,g) H3-R9 multicast groups, but which clients reside beyond which EdgeRTR is not in the mapping system. The H3ServiceEIDs themselves of-course decrypt and parse actual H3-R15 annotations, they also consider during this the

e

MobilityClientEID credentials to avoid "fake-news", but again these are only temporary EIDs allocated to clients in-order to be able to use the mobility network and not for their basic communications.

8. Acknowledgments

This work is partly funded by the ANR LISP-Lab project #ANR-13-INFR-009 (<https://lisplab.lip6.fr>).

9. IANA Considerations

I. Formal H3 to IPv6 EID mapping

II. State enum fields of H3 tiles:

Field 0x: Traffic Direction {

- 0x - null
- 1x - Lane North
- 2x - Lane North + 30
- 3x - Lane North + 60
- 4x - Lane North + 90
- 5x - Lane North + 120
- 6x - Lane North + 150
- 7x - Lane North + 180
- 8x - Lane North + 210
- 9x - Lane North + 240
- Ax - Lane North + 270
- Bx - Lane North + 300
- Cx - Lane North + 330
- Dx - junction
- Ex - shoulder
- Fx - sidewalk

}

field 1x: Persistent or Structural {

- 0x - null
- 1x - pothole light
- 2x - pothole severe
- 3x - speed-bump low
- 4x - speed-bump high
- 5x - icy
- 6x - flooded
- 7x - snow-cover
- 8x - snow-deep
- 9x - construction cone
- Ax - gravel
- Bx - choppy
- Cx - blind-curve
- Dx - steep-slope
- Ex - low-bridge

}

field 2x: Transient Condition {

- 0x - null
- 1x - pedestrian
- 2x - bike scooter
- 3x - stopped car / truck
- 4x - moving car / truck
- 5x - first responder vehicle
- 6x - sudden slowdown
- 7x - oversized over-height vehicle
- 8x - red-light-breach
- 9x - light collision (fender bender)
- Ax - hard collision / casualty
- Bx - collision course car/structure

```

Cx - recent collision residues
Dx - hard brake
Ex - sharp cornering
Fx - freeing-parking
}

field 3x: Traffic-light Cycle {
  0x - null
  1x - 1 seconds to green
  2x - 2 seconds to green
  3x - 3 seconds to green
  4x - 4 seconds to green
  5x - 5 seconds to green
  6x - 6 seconds to green
  7x - 7 seconds to green
  8x - 8 seconds to green
  9x - 9 seconds to green
  Ax - 10 seconds or less
  Bx - 20 seconds or less
  Cx - 30 seconds or less
  Dx - 60 seconds or less
  Ex - green now
  Fx - red now
}

field 4x: Impacted tile from neighboring {
  0x - null
  1x - epicenter
  2x - light yellow
  3x - yellow
  4x - light orange
  5x - orange
  6x - light red
  7x - red
  8x - light blue
  9x - blue
  Ax - green
  Bx - light green
}

field 5x: Transient, Cycle, Impacted, Valid for Next{
  0x - null
  1x - 1sec
  2x - 5sec
  3x - 10sec
  4x - 20sec
  5x - 40sec
  6x - 60sec
  7x - 2min
  8x - 3min
  9x - 4min
  Ax - 5min
  Bx - 10min
  Cx - 15min
  Dx - 30min
  Ex - 60min
  Fx - 24hours
}

field 6x: LaneRightsSigns {
  0x - null
  1x - yield
  2x - speedLimit
  3x - straightOnly
  4x - noStraight
  5x - rightOnly
  6x - noRight

```

- 7x - rightStraight
- 8x - leftOnly
- 9x - leftStraight
- Ax - noLeft
- Bx - noUTurn
- Cx - noLeftU
- Dx - bikeLane
- Ex - HOVLane
- Fx - Stop

```
field 7x: MovementSigns {  
0x - null  
1x - keepRight  
2x - keepLeft  
3x - stayInLane  
4x - doNotEnter  
5x - noTrucks  
6x - noBikes  
7x - noPeds  
8x - oneWay  
9x - parking  
Ax - noParking  
Bx - noStandaing  
Cx - noPassing  
Dx - loadingZone  
Ex - railCross  
Fx - schoolZone  
}
```

```
field 8x: CurvesIntersectSigns {  
0x - null  
1x - turnsLeft  
2x - turnsRight  
3x - curvesLeft  
4x - curvesRight  
5x - reversesLeft  
6x - reversesRight  
7x - windingRoad  
8x - hairPin  
9x - pretzelTurn  
Ax - crossRoads  
Bx - crossT  
Cx - crossY  
Dx - circle  
Ex - laneEnds  
Fx - roadNarrows  
}
```

```
field 9x: Current Tile Speed {  
0x - null  
1x - < 5kmh  
2x - < 10kmh  
3x - < 15kmh  
4x - < 20kmh  
5x - < 30kmh  
6x - < 40kmh  
7x - < 50kmh  
8x - < 60kmh  
9x - < 80kmh  
Ax - < 100kmh  
Bx - < 120kmh  
Cx - < 140kmh  
Dx - < 160kmh  
Ex - > 160kmh  
Fx - queuedTraffic  
}
```

```

field Ax: Vehicle / Pedestrian Traffic {
0x - null
1x - probability of ped/vehicle on tile close to 100%, packed
2x - 95%
3x - 90%
4x - 85%
5x - 80%
6x - 70%
7x - 60%
8x - 50%
9x - 40%
Ax - 30%
Bx - 20%
Cx - 15%
Dx - 10%
Ex - 5%
Fx - probability of ped/vehicle on tile close to 0%, empty
}

field Bx - reserved platooning lineup
field Cx - reserved objects of interest
field Dx - reserved
field Ex - reserved
field Fx - reserved

```

10. Normative References

- [I-D.ietf-lisp-rfc6833bis]
Fuller, V., Farinacci, D., and A. Cabellos-Aparicio,
"Locator/ID Separation Protocol (LISP) Control-Plane",
draft-ietf-lisp-rfc6833bis-07 (work in progress), December
2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The
Locator/ID Separation Protocol (LISP)", RFC 6830,
DOI 10.17487/RFC6830, January 2013,
<<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC8378] Farinacci, D., Moreno, V., "Signal-Free Locator/ID Separation
Protocol (LISP) Multicast", RFC8378,
DOI 10.17487/RFC8378, May 2018,
<<https://www.rfc-editor.org/info/rfc8378>>.

Authors' Addresses

Sharon Barkai
Nexar
CA
USA

Email: sbarkai@gmail.com

Bruno Fernandez-Ruiz
Nexar
London

UK

Email: b@getnexar.com

S ZionB

Nexar

Israel

Email: sharon@fermicloud.io

Rotem Tamir

Nexar

Israel

rotem.tamir@getnexar.com

Alberto Rodriguez-Natal

Cisco Systems

170 Tasman Drive

San Jose, CA

USA

Email: natal@cisco.com

Fabio Maino

Cisco Systems

170 Tasman Drive

San Jose, CA

USA

Email: fmaino@cisco.com

Albert Cabellos-Aparicio

Technical University of Catalonia

Barcelona

Spain

Email: acabello@ac.upc.edu

Jordi Paillissé-Vilanova

Technical University of Catalonia

Barcelona

Spain

Email: jordip@ac.upc.edu

Dino Farinacci

lispers.net

San Jose, CA

USA

Email: farinacci@gmail.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 8, 2019

D. Farinacci
lispers.net
C. Cantrell
Nexus
March 7, 2019

A Decent LISP Mapping System (LISP-Decent)
draft-farinacci-lisp-decent-03

Abstract

This draft describes how the LISP mapping system designed to be distributed for scale can also be decentralized for management and trust.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 8, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Definition of Terms	3
3. Overview	4
4. Push-Based Mapping System	5
4.1. Components of a Pushed-Based LISP-Decent xTR	5
4.2. No LISP Protocol Changes	6
4.3. Configuration and Authentication	7
4.4. Core Seed-Group	7
5. Pull-Based Mapping System	9
5.1. Components of a Pulled-Based LISP-Decent xTR	9
5.2. Deployment Example	10
5.3. Management Considerations	11
6. Security Considerations	11
7. IANA Considerations	12
8. References	12
8.1. Normative References	12
8.2. Informative References	13
Appendix A. Acknowledgments	13
Appendix B. Document Change Log	14
B.1. Changes to draft-farinacci-lisp-decent-03	14
B.2. Changes to draft-farinacci-lisp-decent-02	14
B.3. Changes to draft-farinacci-lisp-decent-01	14
B.4. Changes to draft-farinacci-lisp-decent-00	14
Authors' Addresses	14

1. Introduction

The LISP architecture and protocols [RFC6830] introduces two new numbering spaces, Endpoint Identifiers (EIDs) and Routing Locators (RLOCs) which is intended to provide overlay network functionality. To map from EID to a set of RLOCs, a control-plane mapping system are used [RFC6836] [RFC8111]. These mapping systems are distributed in nature in their deployment for scalability but are centrally managed by a third-party entity, namely a Mapping System Provider (MSP). The entities that use the mapping system, such as data-plane xTRs, depend on and trust the MSP. They do not participate in the mapping system other than to register and retrieve information to/from the mapping system [RFC6833].

This document introduces a Decentralized Mapping System (DMS) so the xTRs can participate in the mapping system as well as use it. They can trust each other rather than rely on third-party infrastructure. The xTRs act as Map-Servers to maintain distributed state for scale and reducing attack surface.

2. Definition of Terms

Mapping System Provider (MSP): is an infrastructure service that deploys LISP Map-Resolvers and Map-Servers [RFC6833] and possibly ALT-nodes [RFC6836] or DDT-nodes [RFC8111]. The MSP can be managed by a separate organization other than the one that manages xTRs. This model provides a business separation between who manages and is responsible for the control-plane versus who manages the data-plane overlay service.

Decentralized Mapping System (DMS): is a mapping system entity that is not third-party to the xTR nodes that use it. The xTRs themselves are part of the mapping system. The state of the mapping system is fully distributed, decentralized, and the trust relies on the xTRs that use and participate in their own mapping system.

Pull-Based Mapping System: the mapping system is pull-based meaning that xTRs will lookup and register mappings by algorithmic transformation to locate which Map-Resolvers and Map-Servers are used. It is required that the lookup and registration uses a consistent algorithmic transformation function. Map-Registers are pushed to specific Map-Servers. Map-Requests are external lookups to Map-Resolvers on xTRs that do not participate in the mapping system and internal lookups when they do.

Modulus Value: this value is used in the Pull-Based Mapping System. It defines the number of map-server sets used for the mapping system. The modulus value is used to produce a Name Index used for a DNS lookup.

Name Index: this index value <index> is used in the Pull-Based Mapping System. For a mapping system that is configured with a map-server set of DNS names in the form of <name>.domain.com, the name index is prepended to <name> to form the lookup name <index>.<name>.domain.com. If the Modulus Value is 8, then the name indexes are 0 through 7.

Hash Mask: The Hash Mask is used in the Pull-Based Mapping System. It is a mask value with 1 bits left justified. The mask is used to select what high-order bits of an EID-prefix is used in the hash function.

Push-Based Mapping System: the mapping system is push-based meaning that xTRs will push registrations via IP multicast to a group of Map-Servers and do local lookups acting as their own Map-Resolvers.

Replication List Entry (RLE): is an RLOC-record format that contains a list of RLOCs that an ITR replicates multicast packets on a multicast overlay. The RLE format is specified in [RFC8060].

RLEs are used with the Pushed-Based mapping system.

Group Address EID: is an EID-record format that contains IPv4 (0.0.0.0/0, G) or IPv6 (0::/0, G) state. This state is encoded as a Multicast Info Type LCAF specified in [RFC8060]. Members of a seed-group send Map-Registers for (0.0.0.0/0, G) or (0::/0, G) with an RLOC-record that RLE encodes its RLOC address. Details are specified in [RFC8378].

Seed-Group: is a set of Map-Servers joined to a multicast group for the Push-Based Mapping system or are mapped by DNS names in a Pull-Based Mapping System. A core seed-group is used to bootstrap a set of LISP-Decent xTRs so they can learn about each other and use each other's mapping system service. A seed-group can be pull-based to bootstrap a push-based mapping system. That is, a set of DNS mapped map-servers can be used to join the mapping system's IP multicast group.

3. Overview

The clients of the Decentralized Mapping System (DMS) are also the providers of mapping state. Clients are typically ETRs that Map-Register EID-to-RLOC mapping state to the mapping database system. ITRs are clients in that they send Map-Requests to the mapping database system to obtain EID-to-RLOC mappings that are cached for data-plane use. When xTRs participate in a DMS, they are also acting as Map-Resolvers and Map-Servers using the protocol machinery defined in LISP control-plane specifications [RFC6833], [I-D.ietf-lisp-sec], and [I-D.ietf-lisp-ecdsa-auth]. The xTRs are not required to run the database mapping transport system protocols specified in [RFC6836] or [RFC8111].

This document will describe two decentralized and distributed mapping system mechanisms. A Push-Based Mapping System uses IP multicast so xTRs can find each other by locally joining an IP multicast group. A Pull-Based Mapping System uses DNS with an algorithmic transformation function so xTRs can find each other.

4. Push-Based Mapping System

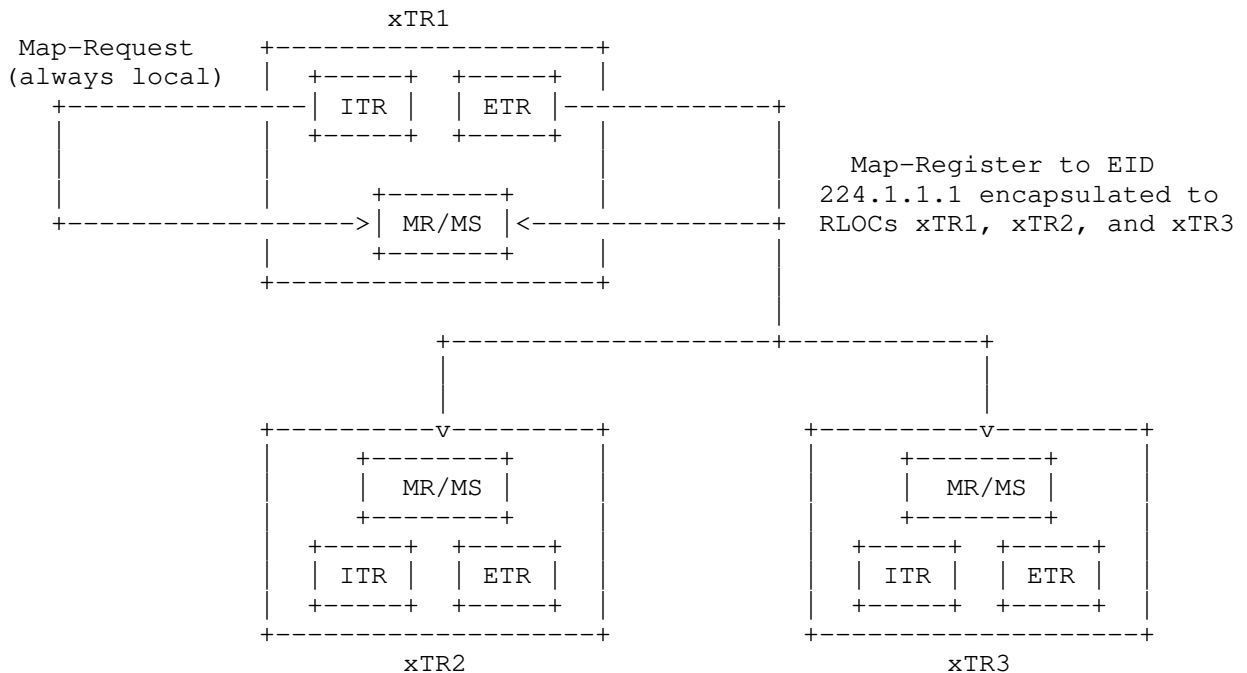
The xTRs are organized in a mapping-system group. The group is identified by an IPv4 or IPv6 multicast group address or using a pull-based approach in described in Section 5. When using multicast, the xTRs join the same multicast group and receive LISP control-plane messages addressed to the group. Messages sent to the multicast group are distributed when the underlay network supports IP multicast [RFC6831] or is achieved with the overlay multicast mechanism described in [RFC8378]. When overlay multicast is used and LISP Map-Register messages are sent to the group, they are LISP data encapsulated with a instance-ID set to 0xffffffff in the LISP header. The inner header of the encapsulated packet has the destination address set to the multicast group address and the outer header that is prepended has the destination address set to the RLOC of mapping system member. The members of the mapping system group are kept in the LISP data-plane map-cache so packets for the group can be replicated to each member RLOC.

All xTRs in a mapping system group will store the same registered mappings and maintain the state as Map-Servers normally do. The members are not only receivers of the multicast group but also send packets to the group.

4.1. Components of a Pushed-Based LISP-Decent xTR

When an xTR is configured to be a LISP-Decent xTR (or PxTR [RFC6832]), it runs the ITR, ETR, Map-Resolver, and Map-Server LISP network functions.

The following diagram shows 3 LISP-Decent xTRs joined to mapping system group 224.1.1.1. When the ETR function of xTR1 originates a Map-Register, it is sent to all xTRs (including itself) synchronizing all 3 Map-Servers in xTR1, xTR2, and xTR3. The ITR function can populate its map-cache by sending a Map-Request locally to its Map-Resolver so it can replicate packets to each RLOC for EID 224.1.1.1.



Note if any external xTR would like to use a Map-Resolver from the mapping system group, it only needs to have one of the LISP-Decent Map-Resolvers configured. By doing a looking to this Map-Resolver for EID 224.1.1.1, the external xTR could get the complete list of members for the mapping system group.

For future study, an external xTR could multicast the Map-Request to 224.1.1.1 and either one of the LISP-Decent Map-Resolvers would return a Map-Reply or the external xTR is prepared to receive multiple Map-Replies.

4.2. No LISP Protocol Changes

There are no LISP protocol changes required to support the push-based LISP-Decent set of procedures. However, an implementation that sends Map-Register messages to a multicast group versus a specific Map-Server unicast address must change to call the data-plane component so the ITR functionality in the node can encapsulate the Map-Register as a unicast packet to each member of the mapping system group.

An ITR SHOULD lookup its mapping system group address periodically to determine if the membership has changed. The ITR can also use the

pubsub capability documented in [I-D.ietf-lisp-pubsub] to be notified when a new member joins or leaves the multicast group.

4.3. Configuration and Authentication

When xTRs are joined to a multicast group, they must have their site registration configuration consistent. Any policy or authentication key material must be configured correctly and consistently among all members. When [I-D.ietf-lisp-ecdsa-auth] is used to sign Map-Register messages, public-keys can be registered to the mapping system group using the site authentication key mentioned above or using a different authentication key from the one used for registering EID records.

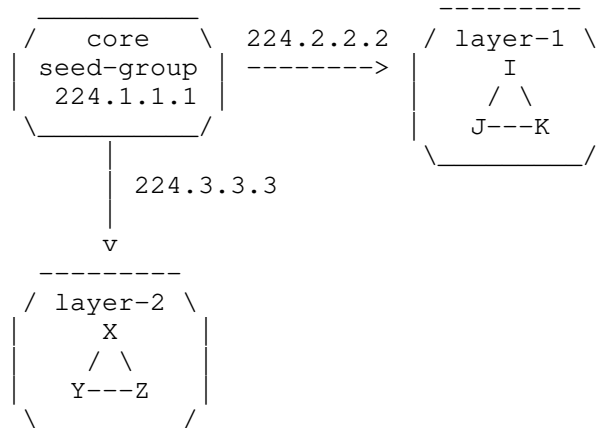
4.4. Core Seed-Group

A core seed-group can be discovered using a multicast group in a push-based system or a Map-Server set of DNS names in a pull-based system (see Section 5 for details).

When using multicast for the mapping system group, a core seed-group multicast group address can be preconfigured to bootstrap the decentralized mapping system. The group address (or DNS name that maps to a group address) can be explicitly configured in a few xTRs to start building up the registrations. Then as other xTRs come online, they can add themselves to the core seed-group by joining the seed-group multicast group.

Alternatively or additionally, new xTRs can join a new mapping system multicast group to form another layer of a decentralized mapping system. The group address and members of this new layer seed-group would be registered to the core seed-group address and stored in the core seed-group mapping system. Note each mapping system layer could have a specific function or a specific circle of trust.

This multi-layer mapping system can be illustrated:



Configured in xTRs A, B, and C (they make up the core seed-group):

224.1.1.1 -> RLE: A, B, C

core seed-group DMS, mapping state in A, B, and C:

224.2.2.2 -> RLE: I, J, K

224.3.3.3 -> RLE: X, Y, Z

layer-1 seed-group DMS (inter-continental), mapping state in I, J, K:

EID1 -> RLOCs: i(1), j(2)

...

EIDn -> RLOCs: i(n), j(n)

layer-2 seed-group DMS (intra-continental), mapping state in X, Y, Z:

EIDa -> RLOCs: x(1), y(2)

...

EIDz -> RLOCs: x(n), y(n)

The core seed-group multicast address 224.1.1.1 is configured in xTRs A, B and C so when each of them send Map-Register messages, they would all be able to maintain synchronized mapping state. Any EID can be registered to this DMS but in this example, seed-group multicast group EIDs are being registered only to find other mapping system groups.

For example, let's say that xTR I boots up and it wants to find its other peers in its mapping system group 224.2.2.2. Group address 224.2.2.2 is configured so xTR I knows what group to join for its mapping system group. But xTR I needs a mapping system to register to, so the core seed-group is used and available to receive Map-

Registers. The other xTRs J and K in the mapping system group do the same so when any of I, J or K needs to register EIDs, they can now send their Map-Register messages to group 224.2.2.2. Examples of EIDs being register are EID1 through EIDn shown above.

When Map-Registers are sent to group 224.2.2.2, they are encapsulated by the LISP data-plane by looking up EID 224.2.2.2 in the core seed-group mapping system. For the map-cache entry to be populated for 224.2.2.2, the data-plane must send a Map-Request so the RLOCs I, J, and K are cached for replication. To use the core seed-group mapping system, the data-plane must know of at least one of the RLOCs A, B, and/or C.

5. Pull-Based Mapping System

5.1. Components of a Pulled-Based LISP-Decent xTR

When an xTR is configured to be a LISP-Decent xTR (or PxTR [RFC6832]), it runs the ITR, ETR, Map-Resolver, and Map-Server LISP network functions.

Unlike the Push-Based Mapping System, the xTRs do not need to be organized by joining a multicast group. In a Pull-Based Mapping System, a hash function over an EID is used to identify which xTR is used as the Map-Resolver and Map-Server. The Domain Name System (DNS) [RFC1034] [RFC1035] is used as a resource discovery mechanism.

The RLOC addresses of the xTRs will be A and AAAA records for DNS names that map algorithmically from the hash of the EID. A SHA-256 hash function [RFC6234] over the following ASCII formatted EID string is used:

```
[<iid>]<eid>/<ml>
[<iid>]<group>/<gml>-<source>/<sml>
```

Where <iid> is the instance-ID and <eid> is the EID of any EID-type defined in [RFC8060]. And then the Modulus Value <mv> is used to produce the Name Index <index> used to build the DNS lookup name:

```
eid = "[<iid>]<eid>/<ml>"
index = hash.sha_256(eid) MOD mv
```

The Hash Mask is used to select what bits are used in the SHA-256 hash function. This is required to support longest match lookups in the mapping system. The same map-server set needs to be selected when looking up a more-specific EID found in the Map-Request message with one that could match a less-specific EID-prefix registered and found in the Map-Register message. For example, if an EID-prefix

[0]240.0.1.0/24 is registered to the mapping system and EID
[0]240.0.1.1/32 is looked up to match the registered prefix, a Hash
Mask of 8 bytes can be used to AND both the /32 or /24 entries to
produce the same hash string bits of "[0]240.0".

For (*,G) and (S,G) multicast entries in the mapping system, the hash
strings are:

```
sg-eid = "<iid><group>/<gml>-<source>/<sml>"  
index = hash.sha_256(sg-eid) MOD mv
```

```
starg-eid = "<iid><group>/<gml>-0.0.0.0/0"  
index = hash.sha_256(starg-eid) MOD mv
```

The Hash Mask MUST include the string "<iid><group>" and not string
<source>. So when looking up [0](2.2.2.2, 224.1.1.1) that will match
a (*, 224.1.1.1/32), the hash string produced with a Hash Mask of 12
bytes is "[0]224.1.1.1".

When the <index> is computed from a unicast or multicast EID, the DNS
lookup name becomes:

```
<index>.map-server.domain.com
```

When an xTR does a DNS lookup on the lookup name, it will send Map-
Register messages to all A and AAAA records for EID registrations.
For Map-Request messages, xTRs MAY round robin EID lookup requests
among the A and AAAA records.

5.2. Deployment Example

Here is an example deployment of a pull-based model. Let's say 4
map-server sets are provisioned for the mapping system. Therefore 4
distinct DNS names are allocated and a Modulus Value 4 is used. Each
DNS name is allocated Name Index 0 through 3:

```
0.map-server.lispers.net  
1.map-server.lispers.net  
2.map-server.lispers.net  
3.map-server.lispers.net
```

The A records for each name can be assigned as:

```
0.map-server.lispers.net:
  A <rloc1-att>
  A <rloc2-verizon>
1.map-server.lispers.net:
  A <rloc1-bt>
  A <rloc2-dt>
2.map-server.lispers.net:
  A <rloc1-cn>
  A <rloc2-kr>
3.map-server.lispers.net:
  A <rloc1-au>
  A <rloc2-nz>
```

When an xTR wants to register "[1000]fd::2222", it hashes the EID string to produce, for example, hash value 0x66. Using the modulus value 4 (0x67 & 0x3) produces index 0x3, so the DNS name 3.map-server.lispers.net is used and a Map-Register is sent to <rloc1-au> and <rloc2-nz>.

Note that the pull-based method can be used for a core seed-group for bootstrapping a push-based mapping system where multicast groups are registered.

5.3. Management Considerations

There are no LISP protocol changes required to support the pull-based LISP-Decent set of procedures. However, an implementation SHOULD do periodic DNS lookups to determine if A records have changed for a DNS entry.

When xTRs derive Map-Resolver and Map-Server names from the DNS, they need to use the same Modulus Value otherwise some xTRs will lookup EIDs to the wrong place they were registered.

The Modulus Value can be configured or pushed to the LISP-Decent xTRs. A future version of this document will describe a push mechanism so all xTRs use a consistent modulus value.

6. Security Considerations

Refer to the Security Considerations section of [I-D.ietf-lisp-rfc6833bis] for a complete list of security mechanisms as well as pointers to threat analysis drafts.

7. IANA Considerations

At this time there are no specific requests for IANA.

8. References

8.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC6831] Farinacci, D., Meyer, D., Zwiebel, J., and S. Venaas, "The Locator/ID Separation Protocol (LISP) for Multicast Environments", RFC 6831, DOI 10.17487/RFC6831, January 2013, <<https://www.rfc-editor.org/info/rfc6831>>.
- [RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <<https://www.rfc-editor.org/info/rfc6832>>.
- [RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, DOI 10.17487/RFC6833, January 2013, <<https://www.rfc-editor.org/info/rfc6833>>.
- [RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, DOI 10.17487/RFC6836, January 2013, <<https://www.rfc-editor.org/info/rfc6836>>.

- [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.
- [RFC8111] Fuller, V., Lewis, D., Ermagan, V., Jain, A., and A. Smirnov, "Locator/ID Separation Protocol Delegated Database Tree (LISP-DDT)", RFC 8111, DOI 10.17487/RFC8111, May 2017, <<https://www.rfc-editor.org/info/rfc8111>>.
- [RFC8378] Moreno, V. and D. Farinacci, "Signal-Free Locator/ID Separation Protocol (LISP) Multicast", RFC 8378, DOI 10.17487/RFC8378, May 2018, <<https://www.rfc-editor.org/info/rfc8378>>.

8.2. Informative References

- [I-D.ietf-lisp-ecdsa-auth]
Farinacci, D. and E. Nordmark, "LISP Control-Plane ECDSA Authentication and Authorization", draft-ietf-lisp-ecdsa-auth-00 (work in progress), September 2018.
- [I-D.ietf-lisp-pubsub]
Rodriguez-Natal, A., Ermagan, V., Leong, J., Maino, F., Cabellos-Aparicio, A., Barkai, S., Farinacci, D., Boucadair, M., Jacquenet, C., and S. Secci, "Publish/Subscribe Functionality for LISP", draft-ietf-lisp-pubsub-02 (work in progress), November 2018.
- [I-D.ietf-lisp-rfc6833bis]
Fuller, V., Farinacci, D., and A. Cabellos-Aparicio, "Locator/ID Separation Protocol (LISP) Control-Plane", draft-ietf-lisp-rfc6833bis-24 (work in progress), February 2019.
- [I-D.ietf-lisp-sec]
Maino, F., Ermagan, V., Cabellos-Aparicio, A., and D. Saucez, "LISP-Security (LISP-SEC)", draft-ietf-lisp-sec-17 (work in progress), November 2018.

Appendix A. Acknowledgments

The authors would like to thank the LISP WG for their review and acceptance of this draft.

The authors would also like to give a special thanks to Roman Shaposhnik for several discussions that occurred before the first draft was published.

Appendix B. Document Change Log

[RFC Editor: Please delete this section on publication as RFC.]

B.1. Changes to draft-farinacci-lisp-decent-03

- o Posted March 2019.
- o Introduce the Hash Mask which is used to grab common bits from a registered prefix and a lookup prefix.
- o Spec how multicast lookups are done in the pull-based mapping system.
- o Indicate the hash string includes the unicast EID mask-length and multicast group and source mask-lengths.

B.2. Changes to draft-farinacci-lisp-decent-02

- o Posted November 2018.
- o Changed references from peer-group to seed-group to make the algorithms in this document more like how blockchain networks initialize the peer-to-peer network.
- o Added pull mechanism to compliment the push mechanism. The pull mechanism could be used as a seed-group to bootstrap the push mechanism.

B.3. Changes to draft-farinacci-lisp-decent-01

- o Posted July 2018.
- o Document timer and reference update.

B.4. Changes to draft-farinacci-lisp-decent-00

- o Initial draft posted January 2018.

Authors' Addresses

Dino Farinacci
lispers.net
San Jose, CA
USA

Email: farinacci@gmail.com

Colin Cantrell
Nexus
Tempe, AZ
USA

Email: colin@nexus.io

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: August 25, 2022

D. Farinacci
lispers.net
C. Cantrell
Nexus
February 21, 2022

A Decent LISP Mapping System (LISP-Decent)
draft-farinacci-lisp-decent-09

Abstract

This draft describes how the LISP mapping system designed to be distributed for scale can also be decentralized for management and trust.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 25, 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Definition of Terms	3
3. Overview	4
4. Push-Based Mapping System	5
4.1. Components of a Pushed-Based LISP-Decent xTR	5
4.2. No LISP Protocol Changes	6
4.3. Configuration and Authentication	7
4.4. Core Seed-Group	7
5. Pull-Based Mapping System	9
5.1. Components of a Pulled-Based LISP-Decent xTR	9
5.2. Deployment Example	10
5.3. Management Considerations	11
6. Security Considerations	11
7. IANA Considerations	12
8. References	12
8.1. Normative References	12
8.2. Informative References	13
Appendix A. Acknowledgments	13
Appendix B. Document Change Log	14
B.1. Changes to draft-farinacci-lisp-decent-09	14
B.2. Changes to draft-farinacci-lisp-decent-08	14
B.3. Changes to draft-farinacci-lisp-decent-07	14
B.4. Changes to draft-farinacci-lisp-decent-06	14
B.5. Changes to draft-farinacci-lisp-decent-05	14
B.6. Changes to draft-farinacci-lisp-decent-04	14
B.7. Changes to draft-farinacci-lisp-decent-03	14
B.8. Changes to draft-farinacci-lisp-decent-02	15
B.9. Changes to draft-farinacci-lisp-decent-01	15
B.10. Changes to draft-farinacci-lisp-decent-00	15
Authors' Addresses	15

1. Introduction

The LISP architecture and protocols [RFC6830] introduces two new numbering spaces, Endpoint Identifiers (EIDs) and Routing Locators (RLOCs) which is intended to provide overlay network functionality. To map from EID to a set of RLOCs, a control-plane mapping system are used [RFC6836] [RFC8111]. These mapping systems are distributed in nature in their deployment for scalability but are centrally managed by a third-party entity, namely a Mapping System Provider (MSP). The entities that use the mapping system, such as data-plane xTRs, depend on and trust the MSP. They do not participate in the mapping system other than to register and retrieve information to/from the mapping system [RFC6833].

This document introduces a Decentralized Mapping System (DMS) so the xTRs can participate in the mapping system as well as use it. They can trust each other rather than rely on third-party infrastructure. The xTRs act as Map-Servers to maintain distributed state for scale and reducing attack surface.

2. Definition of Terms

Mapping System Provider (MSP): is an infrastructure service that deploys LISP Map-Resolvers and Map-Servers [RFC6833] and possibly ALT-nodes [RFC6836] or DDT-nodes [RFC8111]. The MSP can be managed by a separate organization other than the one that manages xTRs. This model provides a business separation between who manages and is responsible for the control-plane versus who manages the data-plane overlay service.

Decentralized Mapping System (DMS): is a mapping system entity that is not third-party to the xTR nodes that use it. The xTRs themselves are part of the mapping system. The state of the mapping system is fully distributed, decentralized, and the trust relies on the xTRs that use and participate in their own mapping system.

Pull-Based Mapping System: the mapping system is pull-based meaning that xTRs will lookup and register mappings by algorithmic transformation to locate which Map-Resolvers and Map-Servers are used. It is required that the lookup and registration uses a consistent algorithmic transformation function. Map-Registers are pushed to specific Map-Servers. Map-Requests are external lookups to Map-Resolvers on xTRs that do not participate in the mapping system and internal lookups when they do.

Modulus Value: this value is used in the Pull-Based Mapping System. It defines the number of map-server sets used for the mapping system. The modulus value is used to produce a Name Index used for a DNS lookup.

Name Index: this index value <index> is used in the Pull-Based Mapping System. For a mapping system that is configured with a map-server set of DNS names in the form of <name>.domain.com, the name index is prepended to <name> to form the lookup name <index>.<name>.domain.com. If the Modulus Value is 8, then the name indexes are 0 through 7.

Hash Mask: The Hash Mask is used in the Pull-Based Mapping System. It is a mask value with 1 bits left justified. The mask is used to select what high-order bits of an EID-prefix is used in the hash function.

Push-Based Mapping System: the mapping system is push-based meaning that xTRs will push registrations via IP multicast to a group of Map-Servers and do local lookups acting as their own Map-Resolvers.

Replication List Entry (RLE): is an RLOC-record format that contains a list of RLOCs that an ITR replicates multicast packets on a multicast overlay. The RLE format is specified in [RFC8060]. RLEs are used with the Pushed-Based mapping system.

Group Address EID: is an EID-record format that contains IPv4 (0.0.0.0/0, G) or IPv6 (0::/0, G) state. This state is encoded as a Multicast Info Type LCAF specified in [RFC8060]. Members of a seed-group send Map-Registers for (0.0.0.0/0, G) or (0::/0, G) with an RLOC-record that RLE encodes its RLOC address. Details are specified in [RFC8378].

Seed-Group: is a set of Map-Servers joined to a multicast group for the Push-Based Mapping system or are mapped by DNS names in a Pull-Based Mapping System. A core seed-group is used to bootstrap a set of LISP-Decent xTRs so they can learn about each other and use each other's mapping system service. A seed-group can be pull-based to bootstrap a push-based mapping system. That is, a set of DNS mapped map-servers can be used to join the mapping system's IP multicast group.

3. Overview

The clients of the Decentralized Mapping System (DMS) are also the providers of mapping state. Clients are typically ETRs that Map-Register EID-to-RLOC mapping state to the mapping database system. ITRs are clients in that they send Map-Requests to the mapping database system to obtain EID-to-RLOC mappings that are cached for data-plane use. When xTRs participate in a DMS, they are also acting as Map-Resolvers and Map-Servers using the protocol machinery defined in LISP control-plane specifications [RFC6833], [I-D.ietf-lisp-sec], and [I-D.ietf-lisp-ecdsa-auth]. The xTRs are not required to run the database mapping transport system protocols specified in [RFC6836] or [RFC8111].

This document will describe two decentralized and distributed mapping system mechanisms. A Push-Based Mapping System uses IP multicast so xTRs can find each other by locally joining an IP multicast group. A Pull-Based Mapping System uses DNS with an algorithmic transformation function so xTRs can find each other.

4. Push-Based Mapping System

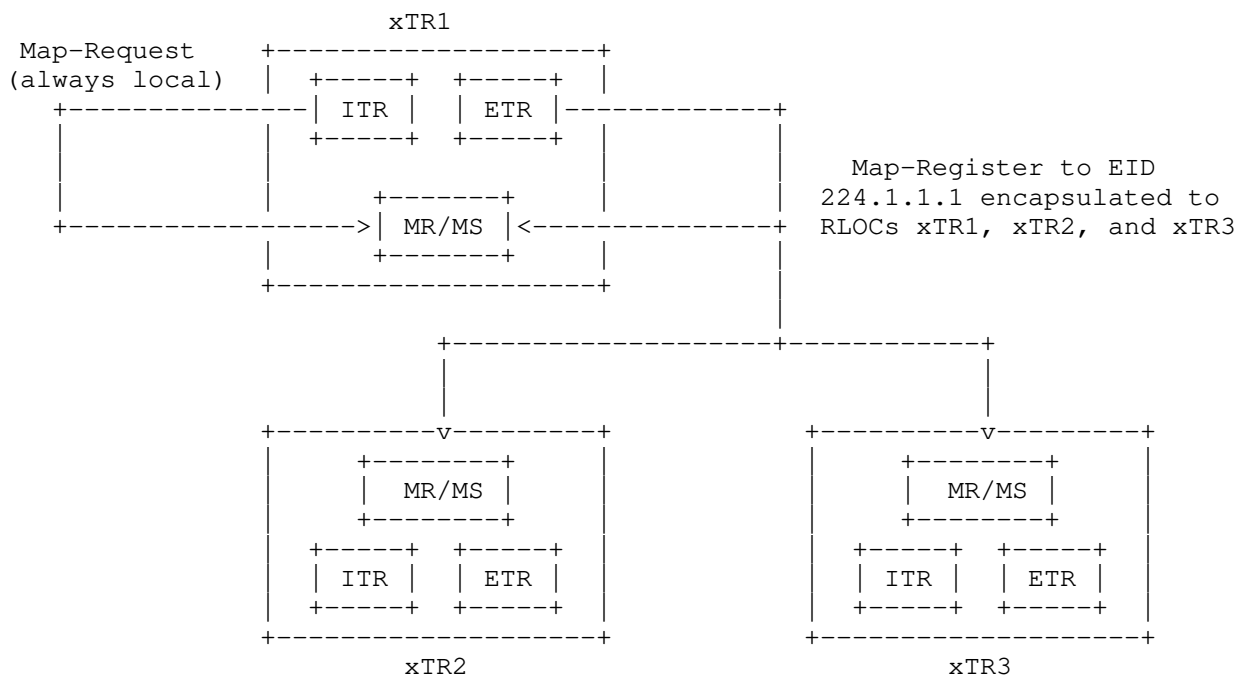
The xTRs are organized in a mapping-system group. The group is identified by an IPv4 or IPv6 multicast group address or using a pull-based approach in described in Section 5. When using multicast, the xTRs join the same multicast group and receive LISP control-plane messages addressed to the group. Messages sent to the multicast group are distributed when the underlay network supports IP multicast [RFC6831] or is achieved with the overlay multicast mechanism described in [RFC8378]. When overlay multicast is used and LISP Map-Register messages are sent to the group, they are LISP data encapsulated with a instance-ID set to 0xffffffff in the LISP header. The inner header of the encapsulated packet has the destination address set to the multicast group address and the outer header that is prepended has the destination address set to the RLOC of mapping system member. The members of the mapping system group are kept in the LISP data-plane map-cache so packets for the group can be replicated to each member RLOC.

All xTRs in a mapping system group will store the same registered mappings and maintain the state as Map-Servers normally do. The members are not only receivers of the multicast group but also send packets to the group.

4.1. Components of a Pushed-Based LISP-Decent xTR

When an xTR is configured to be a LISP-Decent xTR (or PxTR [RFC6832]), it runs the ITR, ETR, Map-Resolver, and Map-Server LISP network functions.

The following diagram shows 3 LISP-Decent xTRs joined to mapping system group 224.1.1.1. When the ETR function of xTR1 originates a Map-Register, it is sent to all xTRs (including itself) synchronizing all 3 Map-Servers in xTR1, xTR2, and xTR3. The ITR function can populate its map-cache by sending a Map-Request locally to its Map-Resolver so it can replicate packets to each RLOC for EID 224.1.1.1.



Note if any external xTR would like to use a Map-Resolver from the mapping system group, it only needs to have one of the LISP-Decent Map-Resolvers configured. By doing a looking to this Map-Resolver for EID 224.1.1.1, the external xTR could get the complete list of members for the mapping system group.

For future study, an external xTR could multicast the Map-Request to 224.1.1.1 and either one of the LISP-Decent Map-Resolvers would return a Map-Reply or the external xTR is prepared to receive multiple Map-Replies.

4.2. No LISP Protocol Changes

There are no LISP protocol changes required to support the push-based LISP-Decent set of procedures. However, an implementation that sends Map-Register messages to a multicast group versus a specific Map-Server unicast address must change to call the data-plane component so the ITR functionality in the node can encapsulate the Map-Register as a unicast packet to each member of the mapping system group.

An ITR SHOULD lookup its mapping system group address periodically to determine if the membership has changed. The ITR can also use the

pubsub capability documented in [I-D.ietf-lisp-pubsub] to be notified when a new member joins or leaves the multicast group.

4.3. Configuration and Authentication

When xTRs are joined to a multicast group, they must have their site registration configuration consistent. Any policy or authentication key material must be configured correctly and consistently among all members. When [I-D.ietf-lisp-ecdsa-auth] is used to sign Map-Register messages, public-keys can be registered to the mapping system group using the site authentication key mentioned above or using a different authentication key from the one used for registering EID records.

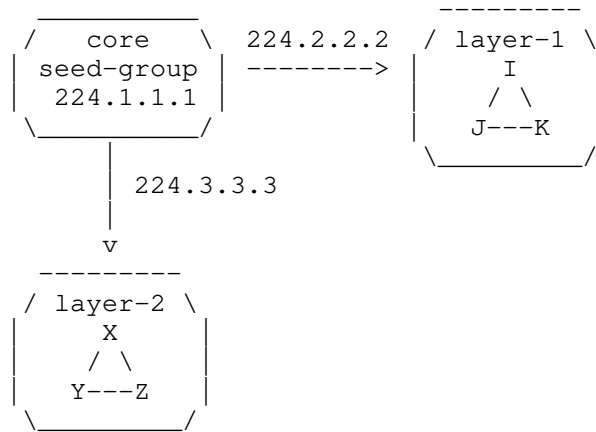
4.4. Core Seed-Group

A core seed-group can be discovered using a multicast group in a push-based system or a Map-Server set of DNS names in a pull-based system (see Section 5 for details).

When using multicast for the mapping system group, a core seed-group multicast group address can be preconfigured to bootstrap the decentralized mapping system. The group address (or DNS name that maps to a group address) can be explicitly configured in a few xTRs to start building up the registrations. Then as other xTRs come online, they can add themselves to the core seed-group by joining the seed-group multicast group.

Alternatively or additionally, new xTRs can join a new mapping system multicast group to form another layer of a decentralized mapping system. The group address and members of this new layer seed-group would be registered to the core seed-group address and stored in the core seed-group mapping system. Note each mapping system layer could have a specific function or a specific circle of trust.

This multi-layer mapping system can be illustrated:



Configured in xTRs A, B, and C (they make up the core seed-group):
 224.1.1.1 -> RLE: A, B, C

core seed-group DMS, mapping state in A, B, and C:

224.2.2.2 -> RLE: I, J, K

224.3.3.3 -> RLE: X, Y, Z

layer-1 seed-group DMS (inter-continental), mapping state in I, J, K:

EID1 -> RLOCs: i(1), j(2)

...

EIDn -> RLOCs: i(n), j(n)

layer-2 seed-group DMS (intra-continental), mapping state in X, Y, Z:

EIDa -> RLOCs: x(1), y(2)

...

EIDz -> RLOCs: x(n), y(n)

The core seed-group multicast address 224.1.1.1 is configured in xTRs A, B and C so when each of them send Map-Register messages, they would all be able to maintain synchronized mapping state. Any EID can be registered to this DMS but in this example, seed-group multicast group EIDs are being registered only to find other mapping system groups.

For example, let's say that xTR I boots up and it wants to find its other peers in its mapping system group 224.2.2.2. Group address 224.2.2.2 is configured so xTR I knows what group to join for its mapping system group. But xTR I needs a mapping system to register to, so the core seed-group is used and available to receive Map-

Registers. The other xTRs J and K in the mapping system group do the same so when any of I, J or K needs to register EIDs, they can now send their Map-Register messages to group 224.2.2.2. Examples of EIDs being register are EID1 through EIDn shown above.

When Map-Registers are sent to group 224.2.2.2, they are encapsulated by the LISP data-plane by looking up EID 224.2.2.2 in the core seed-group mapping system. For the map-cache entry to be populated for 224.2.2.2, the data-plane must send a Map-Request so the RLOCs I, J, and K are cached for replication. To use the core seed-group mapping system, the data-plane must know of at least one of the RLOCs A, B, and/or C.

5. Pull-Based Mapping System

5.1. Components of a Pulled-Based LISP-Decent xTR

When an xTR is configured to be a LISP-Decent xTR (or PxTR [RFC6832]), it runs the ITR, ETR, Map-Resolver, and Map-Server LISP network functions.

Unlike the Push-Based Mapping System, the xTRs do not need to be organized by joining a multicast group. In a Pull-Based Mapping System, a hash function over an EID is used to identify which xTR is used as the Map-Resolver and Map-Server. The Domain Name System (DNS) [RFC1034] [RFC1035] is used as a resource discovery mechanism.

The RLOC addresses of the xTRs will be A and AAAA records for DNS names that map algorithmically from the hash of the EID. A SHA-256 hash function [RFC6234] over the following ASCII formatted EID string is used:

```
[<iid>]<eid>/<ml>
[<iid>]<group>/<gml>-<source>/<sml>
```

Where <iid> is the instance-ID and <eid> is the EID of any EID-type defined in [RFC8060]. And then the Modulus Value <mv> is used to produce the Name Index <index> used to build the DNS lookup name:

```
eid = "[<iid>]<eid>/<ml>"
index = hash.sha_256(eid) MOD mv
```

The Hash Mask is used to select what bits are used in the SHA-256 hash function. This is required to support longest match lookups in the mapping system. The same map-server set needs to be selected when looking up a more-specific EID found in the Map-Request message with one that could match a less-specific EID-prefix registered and found in the Map-Register message. For example, if an EID-prefix

[0]240.0.1.0/24 is registered to the mapping system and EID
[0]240.0.1.1/32 is looked up to match the registered prefix, a Hash
Mask of 8 bytes can be used to AND both the /32 or /24 entries to
produce the same hash string bits of "[0]240.0".

For (*,G) and (S,G) multicast entries in the mapping system, the hash
strings are:

```
sg-eid = "<[iid]><group>/<gml>-<source>/<sml>"
index = hash.sha_256(sg-eid) MOD mv

starg-eid = "<[iid]><group>/<gml>-0.0.0.0/0"
index = hash.sha_256(starg-eid) MOD mv
```

The Hash Mask MUST include the string "<[iid]><group>" and not string
<source>. So when looking up [0](2.2.2.2, 224.1.1.1) that will match
a (*, 224.1.1.1/32), the hash string produced with a Hash Mask of 12
bytes is "[0]224.1.1.1".

When the <index> is computed from a unicast or multicast EID, the DNS
lookup name becomes:

```
<index>.map-server.domain.com
```

When an xTR does a DNS lookup on the lookup name, it will send Map-
Register messages to all A and AAAA records for EID registrations.
For Map-Request messages, xTRs MAY round robin EID lookup requests
among the A and AAAA records.

5.2. Deployment Example

Here is an example deployment of a pull-based model. Let's say 4
map-server sets are provisioned for the mapping system. Therefore 4
distinct DNS names are allocated and a Modulus Value 4 is used. Each
DNS name is allocated Name Index 0 through 3:

```
0.map-server.lispers.net
1.map-server.lispers.net
2.map-server.lispers.net
3.map-server.lispers.net
```

The A records for each name can be assigned as:

```
0.map-server.lispers.net:
  A <rloc1-att>
  A <rloc2-verizon>
1.map-server.lispers.net:
  A <rloc1-bt>
  A <rloc2-dt>
2.map-server.lispers.net:
  A <rloc1-cn>
  A <rloc2-kr>
3.map-server.lispers.net:
  A <rloc1-au>
  A <rloc2-nz>
```

When an xTR wants to register "[1000]fd::2222", it hashes the EID string to produce, for example, hash value 0x66. Using the modulus value 4 (0x67 & 0x3) produces index 0x3, so the DNS name 3.map-server.lispers.net is used and a Map-Register is sent to <rloc1-au> and <rloc2-nz>.

Note that the pull-based method can be used for a core seed-group for bootstrapping a push-based mapping system where multicast groups are registered.

5.3. Management Considerations

There are no LISP protocol changes required to support the pull-based LISP-Decent set of procedures. However, an implementation SHOULD do periodic DNS lookups to determine if A records have changed for a DNS entry.

When xTRs derive Map-Resolver and Map-Server names from the DNS, they need to use the same Modulus Value otherwise some xTRs will lookup EIDs to the wrong place they were registered.

The Modulus Value can be configured or pushed to the LISP-Decent xTRs. A future version of this document will describe a push mechanism so all xTRs use a consistent modulus value.

6. Security Considerations

Refer to the Security Considerations section of [I-D.ietf-lisp-rfc6833bis] for a complete list of security mechanisms as well as pointers to threat analysis drafts.

7. IANA Considerations

At this time there are no specific requests for IANA.

8. References

8.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC6831] Farinacci, D., Meyer, D., Zwiebel, J., and S. Venaas, "The Locator/ID Separation Protocol (LISP) for Multicast Environments", RFC 6831, DOI 10.17487/RFC6831, January 2013, <<https://www.rfc-editor.org/info/rfc6831>>.
- [RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <<https://www.rfc-editor.org/info/rfc6832>>.
- [RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, DOI 10.17487/RFC6833, January 2013, <<https://www.rfc-editor.org/info/rfc6833>>.
- [RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, DOI 10.17487/RFC6836, January 2013, <<https://www.rfc-editor.org/info/rfc6836>>.

- [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.
- [RFC8111] Fuller, V., Lewis, D., Ermagan, V., Jain, A., and A. Smirnov, "Locator/ID Separation Protocol Delegated Database Tree (LISP-DDT)", RFC 8111, DOI 10.17487/RFC8111, May 2017, <<https://www.rfc-editor.org/info/rfc8111>>.
- [RFC8378] Moreno, V. and D. Farinacci, "Signal-Free Locator/ID Separation Protocol (LISP) Multicast", RFC 8378, DOI 10.17487/RFC8378, May 2018, <<https://www.rfc-editor.org/info/rfc8378>>.

8.2. Informative References

- [I-D.ietf-lisp-ecdsa-auth] Farinacci, D. and E. Nordmark, "LISP Control-Plane ECDSA Authentication and Authorization", draft-ietf-lisp-ecdsa-auth-06 (work in progress), August 2021.
- [I-D.ietf-lisp-pubsub] Rodriguez-Natal, A., Ermagan, V., Cabellos, A., Barkai, S., and M. Boucadair, "Publish/Subscribe Functionality for LISP", draft-ietf-lisp-pubsub-09 (work in progress), June 2021.
- [I-D.ietf-lisp-rfc6833bis] Farinacci, D., Maino, F., Fuller, V., and A. Cabellos, "Locator/ID Separation Protocol (LISP) Control-Plane", draft-ietf-lisp-rfc6833bis-30 (work in progress), November 2020.
- [I-D.ietf-lisp-sec] Maino, F., Ermagan, V., Cabellos, A., and D. Saucez, "LISP-Security (LISP-SEC)", draft-ietf-lisp-sec-25 (work in progress), December 2021.

Appendix A. Acknowledgments

The authors would like to thank the LISP WG for their review and acceptance of this draft.

The authors would also like to give a special thanks to Roman Shaposhnik for several discussions that occurred before the first draft was published.

Appendix B. Document Change Log

[RFC Editor: Please delete this section on publication as RFC.]

B.1. Changes to draft-farinacci-lisp-decent-09

- o Posted February 2022.
- o Update references and document expiry timer.

B.2. Changes to draft-farinacci-lisp-decent-08

- o Posted August 2021.
- o Update references and document expiry timer.

B.3. Changes to draft-farinacci-lisp-decent-07

- o Posted March 2021.
- o Update references and document expiry timer.

B.4. Changes to draft-farinacci-lisp-decent-06

- o Posted September 2020.
- o Update references and document expiry timer.

B.5. Changes to draft-farinacci-lisp-decent-05

- o Posted March 2020.
- o Update references and document expiry timer.

B.6. Changes to draft-farinacci-lisp-decent-04

- o Posted September 2019.
- o Update references and document expiry timer.

B.7. Changes to draft-farinacci-lisp-decent-03

- o Posted March 2019.
- o Introduce the Hash Mask which is used to grab common bits from a registered prefix and a lookup prefix.

- o Spec how multicast lookups are done in the pull-based mapping system.
- o Indicate the hash string includes the unicast EID mask-length and multicast group and source mask-lengths.

B.8. Changes to draft-farinacci-lisp-decent-02

- o Posted November 2018.
- o Changed references from peer-group to seed-group to make the algorithms in this document more like how blockchain networks initialize the peer-to-peer network.
- o Added pull mechanism to compliment the push mechanism. The pull mechanism could be used as a seed-group to bootstrap the push mechanism.

B.9. Changes to draft-farinacci-lisp-decent-01

- o Posted July 2018.
- o Document timer and reference update.

B.10. Changes to draft-farinacci-lisp-decent-00

- o Initial draft posted January 2018.

Authors' Addresses

Dino Farinacci
lispers.net
San Jose, CA
USA

Email: farinacci@gmail.com

Colin Cantrell
Nexus
Tempe, AZ
USA

Email: colin@nexus.io

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 21, 2022

D. Farinacci
lispers.net
P. Pillay-Esnault
Independent
W. Haddad
Ericsson
March 20, 2022

LISP EID Anonymity
draft-ietf-lisp-eid-anonymity-12

Abstract

This specification will describe how ephemeral LISP EIDs can be used to create source anonymity. The idea makes use of frequently changing EIDs much like how a credit-card system uses a different credit-card numbers for each transaction.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 21, 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Definition of Terms	3
3. Overview	3
4. Design Details	4
5. Other Types of Ephemeral-EIDs	5
6. Interworking Considerations	5
7. Multicast Considerations	5
8. Performance Improvements	6
9. Security Considerations	6
10. IANA Considerations	6
11. References	6
11.1. Normative References	6
11.2. Informative References	8
Appendix A. Acknowledgments	8
Appendix B. Document Change Log	8
B.1. Changes to draft-ietf-lisp-eid-anonymity-12	8
B.2. Changes to draft-ietf-lisp-eid-anonymity-11	9
B.3. Changes to draft-ietf-lisp-eid-anonymity-10	9
B.4. Changes to draft-ietf-lisp-eid-anonymity-09	9
B.5. Changes to draft-ietf-lisp-eid-anonymity-08	9
B.6. Changes to draft-ietf-lisp-eid-anonymity-07	9
B.7. Changes to draft-ietf-lisp-eid-anonymity-06	9
B.8. Changes to draft-ietf-lisp-eid-anonymity-05	9
B.9. Changes to draft-ietf-lisp-eid-anonymity-04	9
B.10. Changes to draft-ietf-lisp-eid-anonymity-03	10
B.11. Changes to draft-ietf-lisp-eid-anonymity-02	10
B.12. Changes to draft-ietf-lisp-eid-anonymity-01	10
B.13. Changes to draft-ietf-lisp-eid-anonymity-00	10
B.14. Changes to draft-farinacci-lisp-eid-anonymity-02	10
B.15. Changes to draft-farinacci-lisp-eid-anonymity-01	10
B.16. Changes to draft-farinacci-lisp-eid-anonymity-00	11
Authors' Addresses	11

1. Introduction

The LISP architecture [RFC6830] specifies two namespaces, End-Point IDs (EIDs) and Routing Locators (RLOCs). An EID identifies a node in the network and the RLOC indicates the EID's topological location. Typically EIDs are globally unique so an end-node system can connect to any other end-node system on the Internet. Privately used EIDs are allowed when scoped within a VPN but must always be unique within that scope. Therefore, address allocation is required by network administration to avoid address collisions or duplicate address use. In a multiple namespace architecture like LISP, typically the EID will stay fixed while the RLOC can change. This occurs when the EID is mobile or when the LISP site the EID resides in changes its connection to the Internet.

LISP creates the opportunity where EIDs are fixed and won't change. This draft will examine a technique to allow a end-node system to use a temporary address. The lifetime of a temporary address can be the same as a lifetime of an address in use today on the Internet or can have traditionally shorter lifetimes, possibly on the order of a day or even change as frequent as new connection attempts.

2. Definition of Terms

Ephemeral-EID - is an IP address that is created randomly for use for a temporary period of time. An Ephemeral-EID has all the properties of an EID as defined in [RFC6830]. Ephemeral-EIDs are not stored in the Domain Name System (DNS) and should not be used in long-term address referrals.

Client End-Node - is a network node that originates and consumes packets. It is a system that originates packets or initiates the establishment of transport-layer connections. It does not offer services as a server system would. It accesses servers and attempts to do it anonymously.

3. Overview

A client end-node can assign its own ephemeral EID and use it to talk to any system on the Internet. The system is acting as a client where it initiates communication and desires to be an inaccessible resource from any other system. The ephemeral EID is used as a destination address solely to return packets to resources the ephemeral EID connects to. A client-node may simultaneously use a traditional EID along with ephemeral EIDs in parallel and are not mutually exclusive. A client may choose to use the ephemeral EIDs with some peers only where it needs to preserve anonymity.

Here is the procedure a client end-node would use:

1. Client end-node desires to talk on the network. It creates and assigns an ephemeral-EID on any interface. The client end-node may also assign multiple ephemeral-EIDs on the same interface or across different interfaces.
2. If the client end-node is a LISP xTR, it will register ephemeral-EIDs mapped to underlay routable RLOCs. If the client end-node is not a LISP xTR, it can send packets on the network where a LISP router xTR will register the ephemeral-EIDs with its RLOC-set.
3. The client end-node originates packets with a source address equal to the ephemeral-EID and will receive packets addressed to the ephemeral-EID.
4. When the client end-node decides to stop using an ephemeral-EID, it will deregister it from the mapping system and create and assign a new ephemeral-EID, or decide to configure a static global address, or participate in DHCP to get assigned a leased address.

Note that the ephemeral-EID can be mobile just like any other EID so if it is initially registered to the mapping system with one or more RLOCs, later the RLOC-set can change as the ephemeral-EID roams.

4. Design Details

This specification proposes the use of the experimental LISP EID-block 2001:5::/32 [RFC7954] when IPv6 is used. See IANA Considerations section for a specific sub-block allocation request. When IPv4 is used, the Class E block 240.0.0.0/4 is being proposed.

The client end-node system will use the rest of the host bits to allocate a random number to be used as the ephemeral-EID. The EID can be created manually or via a programatic interface. When the EID address is going to change frequently, it is suggested to use a programatic interface. The probability of address collision is unlikely for IPv6 EIDs but could occur for IPv4 EIDs. A client end-node can create an ephemeral-EID and then look it up in the mapping system to see if it exists. If the EID exists in the mapping system, the client end-node can attempt creation of a new random number for the ephemeral-EID. See Section 8 where ephemeral-EIDs can be preallocated and registered to the mapping system before use.

When the client end-node system is co-located with the RLOC and acts as an xTR, it should register the binding before sending packets.

This eliminates a race condition for returning packets not knowing where to encapsulate packets to the ephemeral-EID's RLOCs. See Section 8 for alternatives for fixing this race condition problem. When the client end-node system is not acting as an xTR, it should send some packets so its ephemeral-EID can be discovered by an xTR which supports EID-mobility [I-D.ietf-lisp-eid-mobility] so mapping system registration can occur before the destination returns packets. When the end-node system is acting as an xTR, the EID and RLOC-set is co-located in the same node. So when the EID is created, the xTR can register the mapping versus waiting for packet transmission.

5. Other Types of Ephemeral-EIDs

When IPv6 Ephemeral-EIDs are used, an alternative to a random number can be used. For example, the low-order bits of the IPv6 address could be a cryptographic hash of a public-key. Mechanisms from [RFC3972] could be used for EIDs. Using this approach allows the sender with a hashed EID to be authenticated. So packet signatures can be verified by the corresponding public-key. When hashed EIDs are used, the EID can change frequently as rekeying may be required for enhanced security. LISP specific control message signature mechanisms can be found in [I-D.farinacci-lisp-ecdsa-auth].

6. Interworking Considerations

If a client end-node is communicating with a system that is not in a LISP site, the procedures from [RFC6832] should be followed. The PITR will be required to originate route advertisements for the ephemeral-EID sub-block [RFC7954] so it can attract packets sourced by non-LISP sites destined to ephemeral-EIDs. However, in the general case, the coarse block from [RFC7954] will be advertised which would cover the sub-block. For IPv4, the 240.0.0.0/4 must be advertised into the IPv4 routing system.

7. Multicast Considerations

A client end-node system can be a member of a multicast group fairly easily since its address is not used for multicast communication as a receiver. This is due to the design characteristics of IGMP [RFC3376] [RFC2236] [RFC1112] and MLD [RFC2710] [RFC3810].

When a client end-node system is a multicast source, there is ephemeral (S,G) state that is created and maintained in the network via multicast routing protocols such as PIM [RFC4602] and when PIM is used with LISP [RFC6802]. In addition, when [I-D.ietf-lisp-signal-free-multicast] is used, ephemeral-EID state is created in the mapping database. This doesn't present any problems

other than the amount of state that may exist in the network if not timed out and removed promptly.

However, there exists a multicast source discovery problem when PIM-SSM [RFC4607] is used. Members that join (S,G) channels via out of band mechanisms. These mechanisms need to support ephemeral-EIDs. Otherwise, PIM-ASM [RFC4602] or PIM-Bidir [RFC5015] will need to be used.

8. Performance Improvements

An optimization to reduce the race condition between registering ephemeral-EIDs and returning packets as well as reducing the probability of ephemeral-EID address collision is to preload the mapping database with a list of ephemeral-EIDs before using them. It comes at the expense of rebinding all of registered ephemeral-EIDs when there is an RLOC change. There is work in progress to consider adding a level of indirection here so a single entry gets the RLOC update and the list of ephemeral-EIDs point to the single entry.

9. Security Considerations

When LISP-crypto [RFC8061] is used the EID payload is more secure through encryption providing EID obfuscation of the ephemeral-EID as well as the global-EID it is communicating with. But the obfuscation only occurs between xTRs. So the randomness of a ephemeral-EID inside of LISP sites provide a new level of privacy.

10. IANA Considerations

This specification is requesting the sub-block 2001:5:ffff::/48 for ephemeral-EID usage.

11. References

11.1. Normative References

- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <<https://www.rfc-editor.org/info/rfc1112>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC2236] Fenner, W., "Internet Group Management Protocol, Version 2", RFC 2236, DOI 10.17487/RFC2236, November 1997, <<https://www.rfc-editor.org/info/rfc2236>>.
- [RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, DOI 10.17487/RFC2710, October 1999, <<https://www.rfc-editor.org/info/rfc2710>>.
- [RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <<https://www.rfc-editor.org/info/rfc3376>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.
- [RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, DOI 10.17487/RFC3972, March 2005, <<https://www.rfc-editor.org/info/rfc3972>>.
- [RFC4602] Pusateri, T., "Protocol Independent Multicast - Sparse Mode (PIM-SM) IETF Proposed Standard Requirements Analysis", RFC 4602, DOI 10.17487/RFC4602, August 2006, <<https://www.rfc-editor.org/info/rfc4602>>.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <<https://www.rfc-editor.org/info/rfc4607>>.
- [RFC5015] Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)", RFC 5015, DOI 10.17487/RFC5015, October 2007, <<https://www.rfc-editor.org/info/rfc5015>>.
- [RFC6802] Baillargeon, S., Flinta, C., and A. Johnsson, "Ericsson Two-Way Active Measurement Protocol (TWAMP) Value-Added Octets", RFC 6802, DOI 10.17487/RFC6802, November 2012, <<https://www.rfc-editor.org/info/rfc6802>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.

- [RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <<https://www.rfc-editor.org/info/rfc6832>>.
- [RFC7954] Iannone, L., Lewis, D., Meyer, D., and V. Fuller, "Locator/ID Separation Protocol (LISP) Endpoint Identifier (EID) Block", RFC 7954, DOI 10.17487/RFC7954, September 2016, <<https://www.rfc-editor.org/info/rfc7954>>.
- [RFC8061] Farinacci, D. and B. Weis, "Locator/ID Separation Protocol (LISP) Data-Plane Confidentiality", RFC 8061, DOI 10.17487/RFC8061, February 2017, <<https://www.rfc-editor.org/info/rfc8061>>.

11.2. Informative References

- [I-D.farinacci-lisp-ecdsa-auth]
Farinacci, D. and E. Nordmark, "LISP Control-Plane ECDSA Authentication and Authorization", draft-farinacci-lisp-ecdsa-auth-03 (work in progress), September 2018.
- [I-D.ietf-lisp-eid-mobility]
Comeras, M. P., Ashtaputre, V., Maino, F., Moreno, V., and D. Farinacci, "LISP L2/L3 EID Mobility Using a Unified Control Plane", draft-ietf-lisp-eid-mobility-09 (work in progress), January 2022.
- [I-D.ietf-lisp-signal-free-multicast]
Moreno, V. and D. Farinacci, "Signal-Free Locator/ID Separation Protocol (LISP) Multicast", draft-ietf-lisp-signal-free-multicast-09 (work in progress), March 2018.

Appendix A. Acknowledgments

The author would like to thank the LISP WG for their review and acceptance of this draft.

Appendix B. Document Change Log

[RFC Editor: Please delete this section on publication as RFC.]

B.1. Changes to draft-ietf-lisp-eid-anonymity-12

- o Posted March 2022.
- o Update document timer and references.

- B.2. Changes to draft-ietf-lisp-eid-anonymity-11
 - o Posted end of September 2021.
 - o Update document timer and references.
- B.3. Changes to draft-ietf-lisp-eid-anonymity-10
 - o Posted end of March 2021.
 - o Update document timer and references.
- B.4. Changes to draft-ietf-lisp-eid-anonymity-09
 - o Posted end of October 2020.
 - o Update document timer and references.
- B.5. Changes to draft-ietf-lisp-eid-anonymity-08
 - o Posted end of April 2020.
 - o Update document timer and references.
- B.6. Changes to draft-ietf-lisp-eid-anonymity-07
 - o Posted end of October 2019.
 - o Update document timer and references.
- B.7. Changes to draft-ietf-lisp-eid-anonymity-06
 - o Posted end of March 2019.
 - o Padma had more basic edits and some clarification text.
- B.8. Changes to draft-ietf-lisp-eid-anonymity-05
 - o Posted March IETF week 2019.
 - o Do not state that ephemeral EIDs make the privacy problem worse.
- B.9. Changes to draft-ietf-lisp-eid-anonymity-04
 - o Posted October 2018 before Bangkok IETF deadline.

- o Made Padma requested changes to refer to ephemeral-EIDs allowed to have many on one interface and can be registered with more than 1 RLOC but one RLOC-set.
- B.10. Changes to draft-ietf-lisp-eid-anonymity-03
- o Posted October 2018.
 - o Update document timer and references.
- B.11. Changes to draft-ietf-lisp-eid-anonymity-02
- o Posted April 2018.
 - o Update document timer and references.
- B.12. Changes to draft-ietf-lisp-eid-anonymity-01
- o Posted October 2017.
 - o Add to section 5 that PKI can be used to authenticate EIDs.
 - o Update references.
- B.13. Changes to draft-ietf-lisp-eid-anonymity-00
- o Posted August 2017.
 - o Made draft-farinacci-lisp-eid-anonymity-02 a LISP working group document.
- B.14. Changes to draft-farinacci-lisp-eid-anonymity-02
- o Posted April 2017.
 - o Added section describing how ephemeral-EIDs can use a public key hash as an alternative to a random number.
 - o Indciate when an EID/RLOC co-located, that the xTR can register the EID when it is configured or changed versus waiting for a packet to be sent as in the EID/RLOC separated case.
- B.15. Changes to draft-farinacci-lisp-eid-anonymity-01
- o Posted October 2016.
 - o Update document timer.

B.16. Changes to draft-farinacci-lisp-eid-anonymity-00

- o Posted April 2016.
- o Initial posting.

Authors' Addresses

Dino Farinacci
lispers.net
San Jose, CA
USA

Email: farinacci@gmail.com

Padma Pillay-Esnault
Independent
Santa Clara, CA
USA

Email: padma.ietf@gmail.com

Wassim Haddad
Ericsson
Santa Clara, CA
USA

Email: wassim.haddad@ericsson.com

Network Working Group
Internet-Draft
Obsoletes: 6830 (if approved)
Intended status: Standards Track
Expires: 8 November 2022

D. Farinacci
lispers.net
V. Fuller
vaf.net Internet Consulting
D. Meyer
1-4-5.net
D. Lewis
Cisco Systems
A. Cabellos (Ed.)
UPC/BarcelonaTech
7 May 2022

The Locator/ID Separation Protocol (LISP)
draft-ietf-lisp-rfc6830bis-38

Abstract

This document describes the Data-Plane protocol for the Locator/ID Separation Protocol (LISP). LISP defines two namespaces, End-point Identifiers (EIDs) that identify end-hosts and Routing Locators (RLOCs) that identify network attachment points. With this, LISP effectively separates control from data, and allows routers to create overlay networks. LISP-capable routers exchange encapsulated packets according to EID-to-RLOC mappings stored in a local Map-Cache.

LISP requires no change to either host protocol stacks or to underlay routers and offers Traffic Engineering, multihoming and mobility, among other features.

This document obsoletes RFC 6830.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 November 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
1.1. Scope of Applicability	4
2. Requirements Notation	5
3. Definition of Terms	5
4. Basic Overview	9
4.1. Deployment on the Public Internet	11
4.2. Packet Flow Sequence	11
5. LISP Encapsulation Details	13
5.1. LISP IPv4-in-IPv4 Header Format	13
5.2. LISP IPv6-in-IPv6 Header Format	14
5.3. Tunnel Header Field Descriptions	15
6. LISP EID-to-RLOC Map-Cache	20
7. Dealing with Large Encapsulated Packets	20
7.1. A Stateless Solution to MTU Handling	21
7.2. A Stateful Solution to MTU Handling	22
8. Using Virtualization and Segmentation with LISP	23
9. Routing Locator Selection	23
10. Routing Locator Reachability	25
10.1. Echo Nonce Algorithm	27
11. EID Reachability within a LISP Site	28
12. Routing Locator Hashing	29
13. Changing the Contents of EID-to-RLOC Mappings	30
13.1. Locator-Status-Bits	30
13.2. Database Map-Versioning	31
14. Multicast Considerations	31
15. Router Performance Considerations	32
16. Security Considerations	33
17. Network Management Considerations	34
18. Changes since RFC 6830	34
19. IANA Considerations	35
19.1. LISP UDP Port Numbers	35
20. References	35

20.1. Normative References	35
20.2. Informative References	37
Appendix A. Acknowledgments	40
Appendix B. Document Change Log	41
B.1. Changes to draft-ietf-lisp-rfc6830bis-38	41
B.2. Changes to draft-ietf-lisp-rfc6830bis-37	41
B.3. Changes to draft-ietf-lisp-rfc6830bis-28	41
B.4. Changes to draft-ietf-lisp-rfc6830bis-27	42
B.5. Changes to draft-ietf-lisp-rfc6830bis-26	42
B.6. Changes to draft-ietf-lisp-rfc6830bis-25	42
B.7. Changes to draft-ietf-lisp-rfc6830bis-24	42
B.8. Changes to draft-ietf-lisp-rfc6830bis-23	42
B.9. Changes to draft-ietf-lisp-rfc6830bis-22	43
B.10. Changes to draft-ietf-lisp-rfc6830bis-21	43
B.11. Changes to draft-ietf-lisp-rfc6830bis-20	43
B.12. Changes to draft-ietf-lisp-rfc6830bis-19	43
B.13. Changes to draft-ietf-lisp-rfc6830bis-18	43
B.14. Changes to draft-ietf-lisp-rfc6830bis-17	43
B.15. Changes to draft-ietf-lisp-rfc6830bis-16	43
B.16. Changes to draft-ietf-lisp-rfc6830bis-15	44
B.17. Changes to draft-ietf-lisp-rfc6830bis-14	44
B.18. Changes to draft-ietf-lisp-rfc6830bis-13	44
B.19. Changes to draft-ietf-lisp-rfc6830bis-12	44
B.20. Changes to draft-ietf-lisp-rfc6830bis-11	44
B.21. Changes to draft-ietf-lisp-rfc6830bis-10	44
B.22. Changes to draft-ietf-lisp-rfc6830bis-09	45
B.23. Changes to draft-ietf-lisp-rfc6830bis-08	45
B.24. Changes to draft-ietf-lisp-rfc6830bis-07	45
B.25. Changes to draft-ietf-lisp-rfc6830bis-06	45
B.26. Changes to draft-ietf-lisp-rfc6830bis-05	46
B.27. Changes to draft-ietf-lisp-rfc6830bis-04	46
B.28. Changes to draft-ietf-lisp-rfc6830bis-03	46
B.29. Changes to draft-ietf-lisp-rfc6830bis-02	46
B.30. Changes to draft-ietf-lisp-rfc6830bis-01	47
B.31. Changes to draft-ietf-lisp-rfc6830bis-00	47
Authors' Addresses	47

1. Introduction

This document describes the Locator/Identifier Separation Protocol (LISP). LISP is an encapsulation protocol built around the fundamental idea of separating the topological location of a network attachment point from the node's identity [CHIAPPA]. As a result LISP creates two namespaces: Endpoint Identifiers (EIDs), that are used to identify end-hosts (e.g., nodes or Virtual Machines) and routable Routing Locators (RLOCs), used to identify network attachment points. LISP then defines functions for mapping between the two namespaces and for encapsulating traffic originated by

devices using non-routable EIDs for transport across a network infrastructure that routes and forwards using RLOCs. LISP encapsulation uses a dynamic form of tunneling where no static provisioning is required or necessary.

LISP is an overlay protocol that separates control from Data-Plane, this document specifies the Data-Plane as well as how LISP-capable routers (Tunnel Routers) exchange packets by encapsulating them to the appropriate location. Tunnel routers are equipped with a cache, called Map-Cache, that contains EID-to-RLOC mappings. The Map-Cache is populated using the LISP Control-Plane protocol [I-D.ietf-lisp-rfc6833bis].

LISP does not require changes to either the host protocol stack or to underlay routers. By separating the EID from the RLOC space, LISP offers native Traffic Engineering, multihoming and mobility, among other features.

Creation of LISP was initially motivated by discussions during the IAB-sponsored Routing and Addressing Workshop held in Amsterdam in October 2006 (see [RFC4984]).

This document specifies the LISP Data-Plane encapsulation and other LISP forwarding node functionality while [I-D.ietf-lisp-rfc6833bis] specifies the LISP control plane. LISP deployment guidelines can be found in [RFC7215] and [RFC6835] describes considerations for network operational management. Finally, [I-D.ietf-lisp-introduction] describes the LISP architecture.

This document obsoletes RFC 6830.

1.1. Scope of Applicability

LISP was originally developed to address the Internet-wide route scaling problem [RFC4984]. While there are a number of approaches of interest for that problem, as LISP has been developed and refined, a large number of other LISP uses have been found and are being used. As such, the design and development of LISP has changed so as to focus on these use cases. The common property of these uses is a large set of cooperating entities seeking to communicate over the public Internet or other large underlay IP infrastructures, while keeping the addressing and topology of the cooperating entities separate from the underlay and Internet topology, routing, and addressing.

2. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Definition of Terms

Address Family Identifier (AFI): AFI is a term used to describe an address encoding in a packet. An address family that pertains to addresses found in Data-Plane headers. See [AFN] and [RFC3232] for details. An AFI value of 0 used in this specification indicates an unspecified encoded address where the length of the address is 0 octets following the 16-bit AFI value of 0.

Anycast Address: Anycast Address refers to the same IPv4 or IPv6 address configured and used on multiple systems at the same time. An EID or RLOC can be an anycast address in each of their own address spaces.

Client-side: Client-side is a term used in this document to indicate a connection initiation attempt by an end-system represented by an EID.

Egress Tunnel Router (ETR): An ETR is a router that accepts an IP packet where the destination address in the "outer" IP header is one of its own RLOCs. The router strips the "outer" header and forwards the packet based on the next IP header found. In general, an ETR receives LISP-encapsulated IP packets from the Internet on one side and sends decapsulated IP packets to site end-systems on the other side. ETR functionality does not have to be limited to a router device. A server host can be the endpoint of a LISP tunnel as well.

EID-to-RLOC Database: The EID-to-RLOC Database is a distributed database that contains all known EID-Prefix-to-RLOC mappings. Each potential ETR typically contains a small piece of the database: the EID-to-RLOC mappings for the EID-Prefixes "behind" the router. These map to one of the router's own IP addresses that are routable on the underlay. Note that there MAY be transient conditions when the EID-Prefix for the LISP site and Locator-Set for each EID-Prefix may not be the same on all ETRs. This has no negative implications, since a partial set of Locators can be used.

EID-to-RLOC Map-Cache: The EID-to-RLOC Map-Cache is generally short-

lived, on-demand table in an ITR that stores, tracks, and is responsible for timing out and otherwise validating EID-to-RLOC mappings. This cache is distinct from the full "database" of EID-to-RLOC mappings; it is dynamic, local to the ITR(s), and relatively small, while the database is distributed, relatively static, and much more widely scoped to LISP nodes.

EID-Prefix: An EID-Prefix is a power-of-two block of EIDs that are allocated to a site by an address allocation authority. EID-Prefixes are associated with a set of RLOC addresses. EID-Prefix allocations can be broken up into smaller blocks when an RLOC set is to be associated with the larger EID-Prefix block.

End-System: An end-system is an IPv4 or IPv6 device that originates packets with a single IPv4 or IPv6 header. The end-system supplies an EID value for the destination address field of the IP header when communicating outside of its routing domain. An end-system can be a host computer, a switch or router device, or any network appliance.

Endpoint ID (EID): An EID is a 32-bit (for IPv4) or 128-bit (for IPv6) value that identifies a host. EIDs are generally only found in the source and destination address fields of the first (most inner) LISP header of a packet. The host obtains a destination EID the same way it obtains a destination address today, for example, through a Domain Name System (DNS) [RFC1034] lookup or Session Initiation Protocol (SIP) [RFC3261] exchange. The source EID is obtained via existing mechanisms used to set a host's "local" IP address. An EID used on the public Internet MUST have the same properties as any other IP address used in that manner; this means, among other things, that it MUST be unique. An EID is allocated to a host from an EID-Prefix block associated with the site where the host is located. An EID can be used by a host to refer to other hosts. Note that EID blocks MAY be assigned in a hierarchical manner, independent of the network topology, to facilitate scaling of the mapping database. In addition, an EID block assigned to a site MAY have site-local structure (subnetting) for routing within the site; this structure is not visible to the underlay routing system. In theory, the bit string that represents an EID for one device can represent an RLOC for a different device. When used in discussions with other Locator/ID separation proposals, a LISP EID will be called an "LEID". Throughout this document, any references to "EID" refer to an LEID.

Ingress Tunnel Router (ITR): An ITR is a router that resides in a

LISP site. Packets sent by sources inside of the LISP site to destinations outside of the site are candidates for encapsulation by the ITR. The ITR treats the IP destination address as an EID and performs an EID-to-RLOC mapping lookup. The router then prepends an "outer" IP header with one of its routable RLOCs (in the RLOC space) in the source address field and the result of the mapping lookup in the destination address field. Note that this destination RLOC may be an intermediate, proxy device that has better knowledge of the EID-to-RLOC mapping closer to the destination EID. In general, an ITR receives IP packets from site end-systems on one side and sends LISP-encapsulated IP packets toward the Internet on the other side.

LISP Header: LISP header is a term used in this document to refer to the outer IPv4 or IPv6 header, a UDP header, and a LISP-specific 8-octet header that follow the UDP header and that an ITR prepends or an ETR strips.

LISP Router: A LISP router is a router that performs the functions of any or all of the following: ITR, ETR, RTR, Proxy-ITR (PITR), or Proxy-ETR (PETR).

LISP Site: LISP site is a set of routers in an edge network that are under a single technical administration. LISP routers that reside in the edge network are the demarcation points to separate the edge network from the core network.

Locator-Status-Bits (LSBs): Locator-Status-Bits are present in the LISP header. They are used by ITRs to inform ETRs about the up/down status of all ETRs at the local site. These bits are used as a hint to convey up/down router status and not path reachability status. The LSBs can be verified by use of one of the Locator reachability algorithms described in Section 10. An ETR MUST rate-limit the action it takes when it detects changes in the Locator-Status-Bits.

Proxy-ETR (PETR): A PETR is defined and described in [RFC6832]. A PETR acts like an ETR but does so on behalf of LISP sites that send packets to destinations at non-LISP sites.

Proxy-ITR (PITR): A PITR is defined and described in [RFC6832]. A PITR acts like an ITR but does so on behalf of non-LISP sites that send packets to destinations at LISP sites.

Recursive Tunneling: Recursive Tunneling occurs when a packet has

more than one LISP IP header. Additional layers of tunneling MAY be employed to implement Traffic Engineering or other re-routing as needed. When this is done, an additional "outer" LISP header is added, and the original RLOCs are preserved in the "inner" header.

Re-Encapsulating Tunneling Router (RTR): An RTR acts like an ETR to remove a LISP header, then acts as an ITR to prepend a new LISP header. This is known as Re-encapsulating Tunneling. Doing this allows a packet to be re-routed by the RTR without adding the overhead of additional tunnel headers. When using multiple mapping database systems, care must be taken to not create re-encapsulation loops through misconfiguration.

Route-Returnability: Route-returnability is an assumption that the underlying routing system will deliver packets to the destination. When combined with a nonce that is provided by a sender and returned by a receiver, this limits off-path data insertion. A route-returnability check is verified when a message is sent with a nonce, another message is returned with the same nonce, and the destination of the original message appears as the source of the returned message.

Routing Locator (RLOC): An RLOC is an IPv4 [RFC0791] or IPv6 [RFC8200] address of an Egress Tunnel Router (ETR). An RLOC is the output of an EID-to-RLOC mapping lookup. An EID maps to zero or more RLOCs. Typically, RLOCs are numbered from blocks that are assigned to a site at each point to which it attaches to the underlay network; where the topology is defined by the connectivity of provider networks. Multiple RLOCs can be assigned to the same ETR device or to multiple ETR devices at a site.

Server-side: Server-side is a term used in this document to indicate that a connection initiation attempt is being accepted for a destination EID.

xTR: An xTR is a reference to an ITR or ETR when direction of data flow is not part of the context description. "xTR" refers to the router that is the tunnel endpoint and is used synonymously with the term "Tunnel Router". For example, "An xTR can be located at the Customer Edge (CE) router" indicates both ITR and ETR functionality at the CE router.

4. Basic Overview

One key concept of LISP is that end-systems operate the same way they do today. The IP addresses that hosts use for tracking sockets and connections, and for sending and receiving packets, do not change. In LISP terminology, these IP addresses are called Endpoint Identifiers (EIDs).

Routers continue to forward packets based on IP destination addresses. When a packet is LISP encapsulated, these addresses are referred to as Routing Locators (RLOCs). Most routers along a path between two hosts will not change; they continue to perform routing/forwarding lookups on the destination addresses. For routers between the source host and the ITR as well as routers from the ETR to the destination host, the destination address is an EID. For the routers between the ITR and the ETR, the destination address is an RLOC.

Another key LISP concept is the "Tunnel Router". A Tunnel Router prepends LISP headers on host-originated packets and strips them prior to final delivery to their destination. The IP addresses in this "outer header" are RLOCs. During end-to-end packet exchange between two Internet hosts, an ITR prepends a new LISP header to each packet, and an ETR strips the new header. The ITR performs EID-to-RLOC lookups to determine the routing path to the ETR, which has the RLOC as one of its IP addresses.

Some basic rules governing LISP are:

- * End-systems only send to addresses that are EIDs. EIDs are typically IP addresses assigned to hosts (other types of EID are supported by LISP, see [RFC8060] for further information). End-systems don't know that addresses are EIDs versus RLOCs but assume that packets get to their intended destinations. In a system where LISP is deployed, LISP routers intercept EID-addressed packets and assist in delivering them across the network core where EIDs cannot be routed. The procedure a host uses to send IP packets does not change.
- * LISP routers mostly deal with Routing Locator addresses. See details in Section 4.2 to clarify what is meant by "mostly".
- * RLOCs are always IP addresses assigned to routers, preferably topologically oriented addresses from provider CIDR (Classless Inter-Domain Routing) blocks.
- * When a router originates packets, it MAY use as a source address either an EID or RLOC. When acting as a host (e.g., when terminating a transport session such as Secure SHell (SSH),

TELNET, or the Simple Network Management Protocol (SNMP)), it MAY use an EID that is explicitly assigned for that purpose. An EID that identifies the router as a host MUST NOT be used as an RLOC; an EID is only routable within the scope of a site. A typical BGP configuration might demonstrate this "hybrid" EID/RLOC usage where a router could use its "host-like" EID to terminate iBGP sessions to other routers in a site while at the same time using RLOCs to terminate eBGP sessions to routers outside the site.

- * Packets with EIDs in them are not expected to be delivered end-to-end in the absence of an EID-to-RLOC mapping operation. They are expected to be used locally for intra-site communication or to be encapsulated for inter-site communication.
- * EIDs MAY also be structured (subnetted) in a manner suitable for local routing within an Autonomous System (AS).

An additional LISP header MAY be prepended to packets by a TE-ITR when re-routing of the path for a packet is desired. A potential use-case for this would be an ISP router that needs to perform Traffic Engineering for packets flowing through its network. In such a situation, termed "Recursive Tunneling", an ISP transit acts as an additional ITR, and the destination RLOC it uses for the new prepended header would be either a TE-ETR within the ISP (along an intra-ISP traffic engineered path) or a TE-ETR within another ISP (an inter-ISP traffic engineered path, where an agreement to build such a path exists).

In order to avoid excessive packet overhead as well as possible encapsulation loops, this document RECOMMENDS that a maximum of two LISP headers can be prepended to a packet. For initial LISP deployments, it is assumed that two headers is sufficient, where the first prepended header is used at a site for Location/Identity separation and the second prepended header is used inside a service provider for Traffic Engineering purposes.

Tunnel Routers can be placed fairly flexibly in a multi-AS topology. For example, the ITR for a particular end-to-end packet exchange might be the first-hop or default router within a site for the source host. Similarly, the ETR might be the last-hop router directly connected to the destination host. Another example, perhaps for a VPN service outsourced to an ISP by a site, the ITR could be the site's border router at the service provider attachment point. Mixing and matching of site-operated, ISP-operated, and other Tunnel Routers is allowed for maximum flexibility.

4.1. Deployment on the Public Internet

Several of the mechanisms in this document are intended for deployment in controlled, trusted environments, and are insecure for use over the public Internet. In particular, on the public internet xTRs:

- * MUST set the N, L, E, and V bits in the LISP header (Section 5.1) to zero.
- * MUST NOT use Locator-Status-Bits and echo-nonce, as described in Section 10 for Routing Locator Reachability. Instead MUST rely solely on control-plane methods.
- * MUST NOT use Gleaning or Locator-Status-Bits and Map-Versioning, as described in Section 13 to update the EID-to-RLOC Mappings. Instead relying solely on control-plane methods.

4.2. Packet Flow Sequence

This section provides an example of the unicast packet flow, including also Control-Plane information as specified in [I-D.ietf-lisp-rfc6833bis]. The example also assumes the following conditions:

- * Source host "host1.abc.example.com" is sending a packet to "host2.xyz.example.com", exactly as it would if the site was not using LISP.
- * Each site is multihomed, so each Tunnel Router has an address (RLOC) assigned from the service provider address block for each provider to which that particular Tunnel Router is attached.
- * The ITR(s) and ETR(s) are directly connected to the source and destination, respectively, but the source and destination can be located anywhere in the LISP site.
- * A Map-Request is sent for an external destination when the destination is not found in the forwarding table or matches a default route. Map-Requests are sent to the mapping database system by using the LISP Control-Plane protocol documented in [I-D.ietf-lisp-rfc6833bis].
- * Map-Replies are sent on the underlying routing system topology using the [I-D.ietf-lisp-rfc6833bis] Control-Plane protocol.

Client host1.abc.example.com wants to communicate with server host2.xyz.example.com:

1. host1.abc.example.com wants to open a TCP connection to host2.xyz.example.com. It does a DNS lookup on host2.xyz.example.com. An A/AAAA record is returned. This address is the destination EID. The locally assigned address of host1.abc.example.com is used as the source EID. An IPv4 or IPv6 packet is built and forwarded through the LISP site as a normal IP packet until it reaches a LISP ITR.
2. The LISP ITR must be able to map the destination EID to an RLOC of one of the ETRs at the destination site. A method to do this is to send a LISP Map-Request, as specified in [I-D.ietf-lisp-rfc6833bis].
3. The mapping system helps forwarding the Map-Request to the corresponding ETR. When the Map-Request arrives at one of the ETRs at the destination site, it will process the packet as a control message.
4. The ETR looks at the destination EID of the Map-Request and matches it against the prefixes in the ETR's configured EID-to-RLOC mapping database. This is the list of EID-Prefixes the ETR is supporting for the site it resides in. If there is no match, the Map-Request is dropped. Otherwise, a LISP Map-Reply is returned to the ITR.
5. The ITR receives the Map-Reply message, parses the message, and stores the mapping information from the packet. This information is stored in the ITR's EID-to-RLOC Map-Cache. Note that the Map-Cache is an on-demand cache. An ITR will manage its Map-Cache in such a way that optimizes for its resource constraints.
6. Subsequent packets from host1.abc.example.com to host2.xyz.example.com will have a LISP header prepended by the ITR using the appropriate RLOC as the LISP header destination address learned from the ETR. Note that the packet MAY be sent to a different ETR than the one that returned the Map-Reply due to the source site's hashing policy or the destination site's Locator-Set policy.
7. The ETR receives these packets directly (since the destination address is one of its assigned IP addresses), checks the validity of the addresses, strips the LISP header, and forwards packets to the attached destination host.
8. In order to defer the need for a mapping lookup in the reverse direction, an ETR can OPTIONALLY create a cache entry that maps the source EID (inner-header source IP address) to the source RLOC (outer-header source IP address) in a received LISP packet.

Such a cache entry is termed a "glean mapping" and only contains a single RLOC for the EID in question. More complete information about additional RLOCs SHOULD be verified by sending a LISP Map-Request for that EID. Both the ITR and the ETR MAY also influence the decision the other makes in selecting an RLOC.

5. LISP Encapsulation Details

Since additional tunnel headers are prepended, the packet becomes larger and can exceed the MTU of any link traversed from the ITR to the ETR. It is RECOMMENDED in IPv4 that packets do not get fragmented as they are encapsulated by the ITR. Instead, the packet is dropped and an ICMP Unreachable/Fragmentation-Needed message is returned to the source.

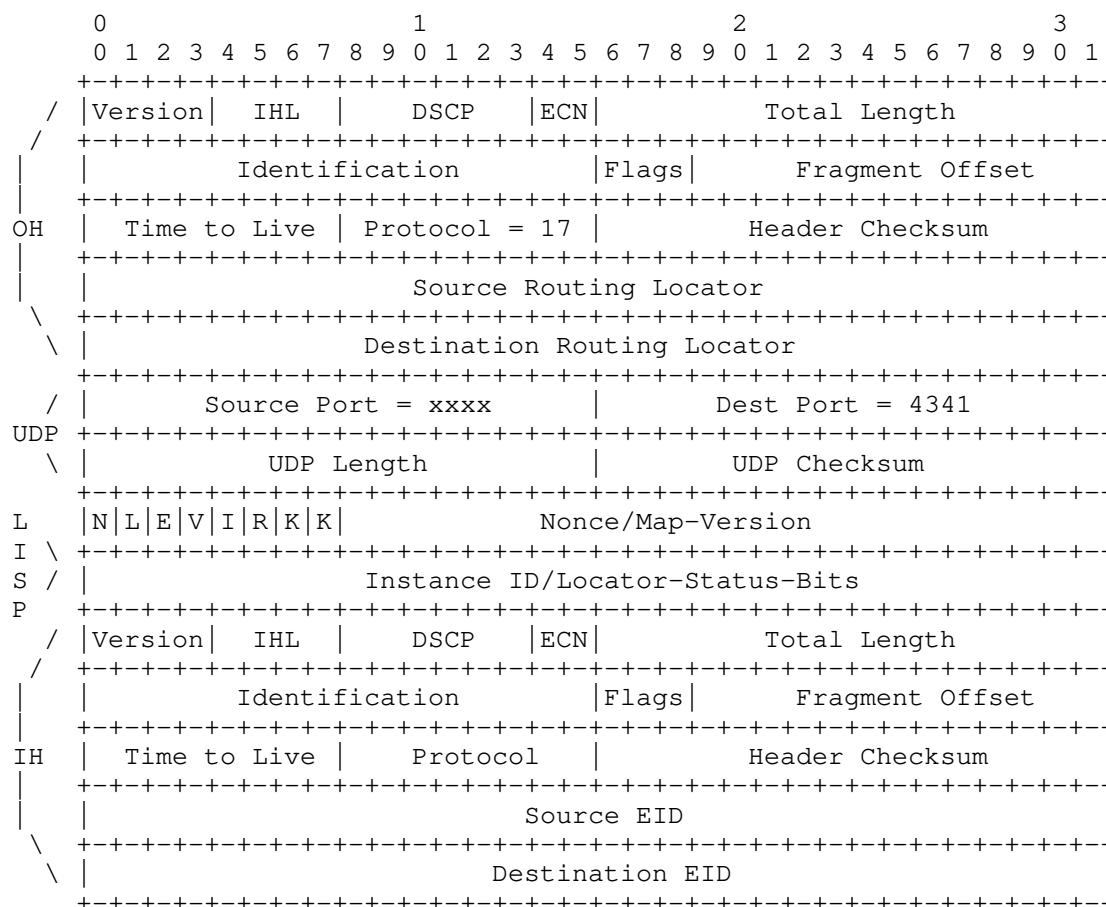
In the case when fragmentation is needed, this specification RECOMMENDS that implementations provide support for one of the proposed fragmentation and reassembly schemes. Two existing schemes are detailed in Section 7.

Since IPv4 or IPv6 addresses can be either EIDs or RLOCs, the LISP architecture supports IPv4 EIDs with IPv6 RLOCs (where the inner header is in IPv4 packet format and the outer header is in IPv6 packet format) or IPv6 EIDs with IPv4 RLOCs (where the inner header is in IPv6 packet format and the outer header is in IPv4 packet format). The next sub-sections illustrate packet formats for the homogeneous case (IPv4-in-IPv4 and IPv6-in-IPv6), but all 4 combinations MUST be supported. Additional types of EIDs are defined in [RFC8060].

As LISP uses UDP encapsulation to carry traffic between xTRs across the Internet, implementors should be aware of the provisions of [RFC8085], especially those given in section 3.1.11 on congestion control for UDP tunneling.

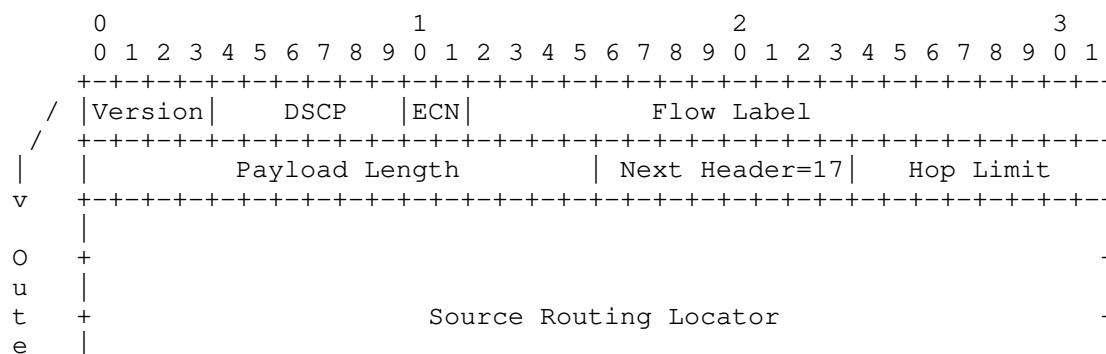
Implementors are encouraged to consider UDP checksum usage guidelines in section 3.4 of [RFC8085] when it is desirable to protect UDP and LISP headers against corruption.

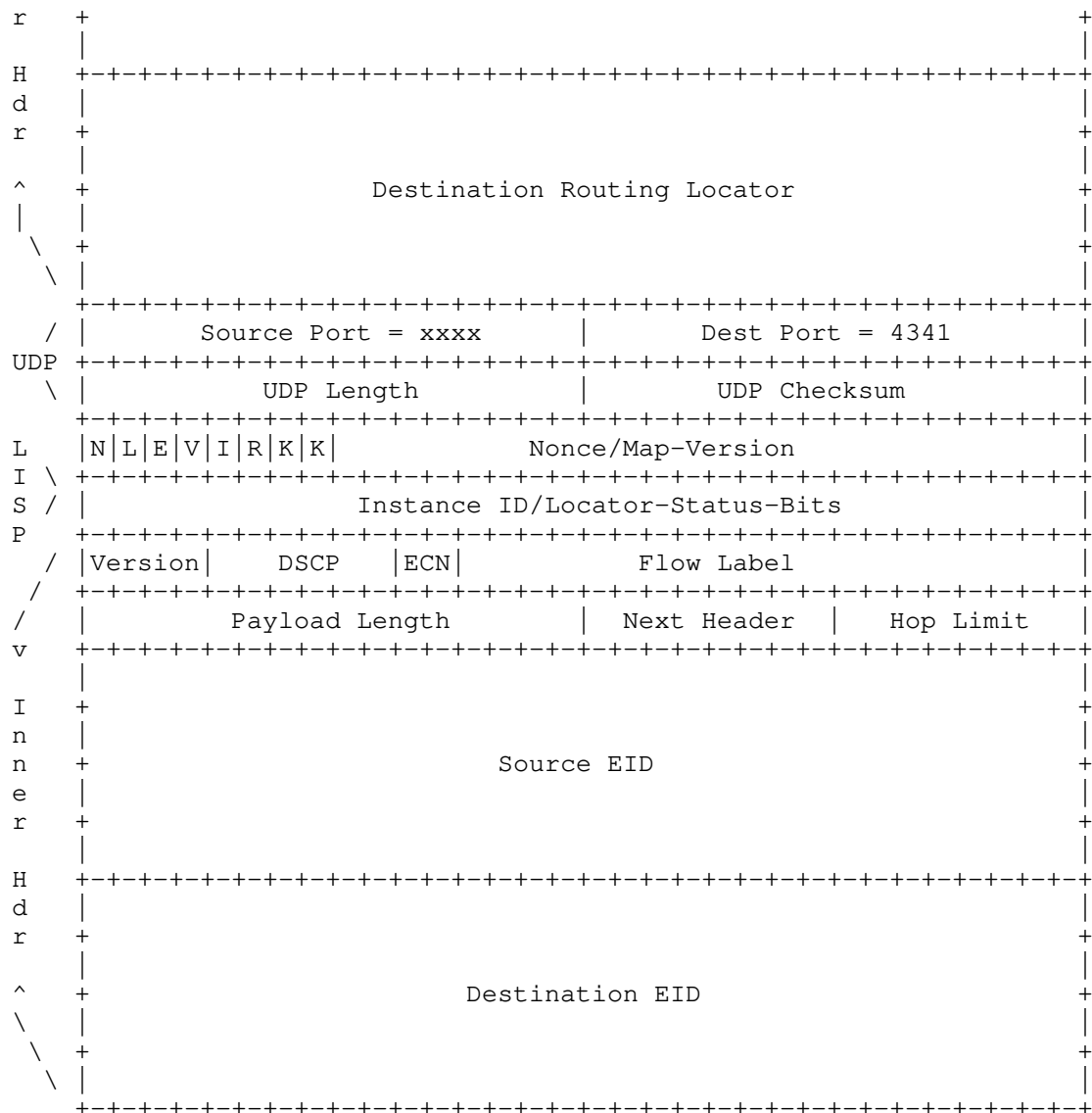
5.1. LISP IPv4-in-IPv4 Header Format



IHL = IP-Header-Length

5.2. LISP IPv6-in-IPv6 Header Format





5.3. Tunnel Header Field Descriptions

Inner Header (IH): The inner header is the header on the datagram received from the originating host [RFC0791] [RFC8200] [RFC2474]. The source and destination IP addresses are EIDs.

Outer Header: (OH) The outer header is a new header prepended by an

ITR. The address fields contain RLOCs obtained from the ingress router's EID-to-RLOC Cache. The IP protocol number is "UDP (17)" from [RFC0768]. The setting of the Don't Fragment (DF) bit 'Flags' field is according to rules listed in Sections 7.1 and 7.2.

UDP Header: The UDP header contains an ITR selected source port when encapsulating a packet. See Section 12 for details on the hash algorithm used to select a source port based on the 5-tuple of the inner header. The destination port MUST be set to the well-known IANA-assigned port value 4341.

UDP Checksum: The 'UDP Checksum' field SHOULD be transmitted as zero by an ITR for either IPv4 [RFC0768] and IPv6 encapsulation [RFC6935] [RFC6936]. When a packet with a zero UDP checksum is received by an ETR, the ETR MUST accept the packet for decapsulation. When an ITR transmits a non-zero value for the UDP checksum, it MUST send a correctly computed value in this field. When an ETR receives a packet with a non-zero UDP checksum, it MAY choose to verify the checksum value. If it chooses to perform such verification, and the verification fails, the packet MUST be silently dropped. If the ETR chooses not to perform the verification, or performs the verification successfully, the packet MUST be accepted for decapsulation. The handling of UDP zero checksums over IPv6 for all tunneling protocols, including LISP, is subject to the applicability statement in [RFC6936].

UDP Length: The 'UDP Length' field is set for an IPv4-encapsulated packet to be the sum of the inner-header IPv4 Total Length plus the UDP and LISP header lengths. For an IPv6-encapsulated packet, the 'UDP Length' field is the sum of the inner-header IPv6 Payload Length, the size of the IPv6 header (40 octets), and the size of the UDP and LISP headers.

N: The N-bit is the nonce-present bit. When this bit is set to 1, the low-order 24 bits of the first 32 bits of the LISP header contain a Nonce. See Section 10.1 for details. Both N- and V-bits MUST NOT be set in the same packet. If they are, a decapsulating ETR MUST treat the 'Nonce/Map-Version' field as having a Nonce value present.

L: The L-bit is the 'Locator-Status-Bits' field enabled bit. When this bit is set to 1, the Locator-Status-Bits in the second 32 bits of the LISP header are in use.

```

      x 1 x x 0 x x x
+-----+
|N|L|E|V|I|R|K|K|      Nonce/Map-Version      |
+-----+
|      Locator-Status-Bits      |
+-----+

```

E: The E-bit is the echo-nonce-request bit. This bit MUST be ignored and has no meaning when the N-bit is set to 0. When the N-bit is set to 1 and this bit is set to 1, an ITR is requesting that the nonce value in the 'Nonce' field be echoed back in LISP-encapsulated packets when the ITR is also an ETR. See Section 10.1 for details.

V: The V-bit is the Map-Version present bit. When this bit is set to 1, the N-bit MUST be 0. Refer to the [I-D.ietf-lisp-6834bis] specification for more details on Database Map-Versioning. This bit indicates that the LISP header is encoded in this case as:

```

      0 x 0 1 x x x x
+-----+
|N|L|E|V|I|R|K|K| Source Map-Version | Dest Map-Version |
+-----+
|      Instance ID/Locator-Status-Bits      |
+-----+

```

I: The I-bit is the Instance ID bit. See Section 8 for more details. When this bit is set to 1, the 'Locator-Status-Bits' field is reduced to 8 bits and the high-order 24 bits are used as an Instance ID. If the I-bit is set to 0, then the low-order 8 bits are transmitted as zero and ignored on receipt. The format of the LISP header would look like this:

```

      x x x x 1 x x x
+-----+
|N|L|E|V|I|R|K|K|      Nonce/Map-Version      |
+-----+
|      Instance ID      |      LSBs      |
+-----+

```

R: The R-bit is a Reserved and unassigned bit for future use. It MUST be set to 0 on transmit and MUST be ignored on receipt.

KK: The KK-bits are a 2-bit field used when encapsulated packets are encrypted. The field is set to 00 when the packet is not encrypted. See [RFC8061] for further information.

LISP Nonce: The LISP 'Nonce' field is a 24-bit value that is

randomly generated by an ITR when the N-bit is set to 1. Nonce generation algorithms are an implementation matter but are required to generate different nonces when sending to different RLOCs. The nonce is also used when the E-bit is set to request the nonce value to be echoed by the other side when packets are returned. When the E-bit is clear but the N-bit is set, a remote ITR is either echoing a previously requested echo-nonce or providing a random nonce. See Section 10.1 for more details. Finally, when both the N and V-bit are not set (N=0, V=0), then both the Nonce and Map-Version fields are set to 0 and ignored on receipt.

LISP Locator-Status-Bits (LSBs): When the L-bit is also set, the 'Locator-Status-Bits' field in the LISP header is set by an ITR to indicate to an ETR the up/down status of the Locators in the source site. Each RLOC in a Map-Reply is assigned an ordinal value from 0 to n-1 (when there are n RLOCs in a mapping entry). The Locator-Status-Bits are numbered from 0 to n-1 from the least significant bit of the field. The field is 32 bits when the I-bit is set to 0 and is 8 bits when the I-bit is set to 1. When a Locator-Status-Bit is set to 1, the ITR is indicating to the ETR that the RLOC associated with the bit ordinal has up status. See Section 10 for details on how an ITR can determine the status of the ETRs at the same site. When a site has multiple EID-Prefixes that result in multiple mappings (where each could have a different Locator-Set), the Locator-Status-Bits setting in an encapsulated packet MUST reflect the mapping for the EID-Prefix that the inner-header source EID address matches (longest-match). If the LSB for an anycast Locator is set to 1, then there is at least one RLOC with that address, and the ETR is considered 'up'.

When doing ITR/PITR encapsulation:

- * The outer-header 'Time to Live' field (or 'Hop Limit' field, in the case of IPv6) SHOULD be copied from the inner-header 'Time to Live' field.
- * The outer-header IPv4 'Differentiated Services Code Point' (DSCP) field or the 'Traffic Class' field, in the case of IPv6, SHOULD be copied from the inner-header IPv4 DSCP field or 'Traffic Class' field in the case of IPv6, to the outer-header. Guidelines for this can be found at [RFC2983].
- * The IPv4 'Explicit Congestion Notification' (ECN) field and bits 6 and 7 of the IPv6 'Traffic Class' field requires special treatment in order to avoid discarding indications of congestion as specified in [RFC6040].

When doing ETR/PETR decapsulation:

- * The inner-header IPv4 'Time to Live' field or 'Hop Limit' field in the case of IPv6, MUST be copied from the outer-header 'Time to Live'/'Hop Limit' field, when the 'Time to Live'/'Hop Limit' value of the outer header is less than the 'Time to Live'/'Hop Limit' value of the inner header. Failing to perform this check can cause the 'Time to Live'/'Hop Limit' of the inner header to increment across encapsulation/decapsulation cycles. This check is also performed when doing initial encapsulation, when a packet comes to an ITR or PITR destined for a LISP site.
- * The outer-header IPv4 'Differentiated Services Code Point' (DSCP) field or the 'Traffic Class' field in the case of IPv6, SHOULD be copied from the outer-header IPv4 DSCP field or 'Traffic Class' field in the case of IPv6, to the inner-header. Guidelines for this can be found at [RFC2983].
- * The IPv4 'Explicit Congestion Notification' (ECN) field and bits 6 and 7 of the IPv6 'Traffic Class' field, requires special treatment in order to avoid discarding indications of congestion as specified in [RFC6040]. Note that implementations exist that copy the 'ECN' field from the outer header to the inner header even though [RFC6040] does not recommend this behavior. It is RECOMMENDED that implementations change to support the behavior in [RFC6040].

Note that if an ETR/PETR is also an ITR/PITR and chooses to re-encapsulate after decapsulating, the net effect of this is that the new outer header will carry the same Time to Live as the old outer header minus 1.

Copying the Time to Live (TTL) serves two purposes: first, it preserves the distance the host intended the packet to travel; second, and more importantly, it provides for suppression of looping packets in the event there is a loop of concatenated tunnels due to misconfiguration.

Some xTRs and PxTRs performs re-encapsulation operations and need to treat the 'Explicit Congestion Notification' (ECN) in a special way. Because the re-encapsulation operation is a sequence of two operations, namely a decapsulation followed by an encapsulation, the ECN bits MUST be treated as described above for these two operations.

The LISP dataplane protocol is not backwards compatible with [RFC6830] and does not have explicit support for introducing future protocol changes (e.g. an explicit version field). However, the LISP control plane [I-D.ietf-lisp-rfc6833bis] allows an ETR to register

dataplane capabilities by means of new LCAF types [RFC8060]. In this way an ITR can be made aware of the dataplane capabilities of an ETR, and encapsulate accordingly. The specification of the new LCAF types, new LCAF mechanisms, and their use, is out of the scope of this document.

6. LISP EID-to-RLOC Map-Cache

ITRs and PITRs maintain an on-demand cache, referred as LISP EID-to-RLOC Map-Cache, that contains mappings from EID-prefixes to locator sets. The cache is used to encapsulate packets from the EID space to the corresponding RLOC network attachment point.

When an ITR/PITR receives a packet from inside of the LISP site to destinations outside of the site a longest-prefix match lookup of the EID is done to the Map-Cache.

When the lookup succeeds, the Locator-Set retrieved from the Map-Cache is used to send the packet to the EID's topological location.

If the lookup fails, the ITR/PITR needs to retrieve the mapping using the LISP Control-Plane protocol [I-D.ietf-lisp-rfc6833bis]. While the mapping is being retrieved, the ITR/PITR can either drop or buffer the packets. This document does not have specific recommendations about the action to be taken. It is up to the deployer to consider whether or not it is desirable to buffer packets and deploy a LISP implementation that offers the desired behaviour. Once the mapping is resolved it is then stored in the local Map-Cache to forward subsequent packets addressed to the same EID-prefix.

The Map-Cache is a local cache of mappings, entries are expired based on the associated Time to live. In addition, entries can be updated with more current information, see Section 13 for further information on this. Finally, the Map-Cache also contains reachability information about EIDs and RLOCs, and uses LISP reachability information mechanisms to determine the reachability of RLOCs, see Section 10 for the specific mechanisms.

7. Dealing with Large Encapsulated Packets

This section proposes two mechanisms to deal with packets that exceed the path MTU between the ITR and ETR.

It is left to the implementor to decide if the stateless or stateful mechanism SHOULD be implemented. Both or neither can be used, since it is a local decision in the ITR regarding how to deal with MTU issues, and sites can interoperate with differing mechanisms.

Both stateless and stateful mechanisms also apply to Re-encapsulating and Recursive Tunneling, so any actions below referring to an ITR also apply to a TE-ITR.

7.1. A Stateless Solution to MTU Handling

An ITR stateless solution to handle MTU issues is described as follows:

1. Define H to be the size, in octets, of the outer header an ITR prepends to a packet. This includes the UDP and LISP header lengths.
2. Define L to be the size, in octets, of the maximum-sized packet an ITR can send to an ETR without the need for the ITR or any intermediate routers to fragment the packet. The network administrator of the LISP deployment has to determine what is the suitable value of L so to make sure that no MTU issues arise.
3. Define an architectural constant S for the maximum size of a packet, in octets, an ITR MUST receive from the source so the effective MTU can be met. That is, $L = S + H$.

When an ITR receives a packet from a site-facing interface and adds H octets worth of encapsulation to yield a packet size greater than L octets (meaning the received packet size was greater than S octets from the source), it resolves the MTU issue by first splitting the original packet into 2 equal-sized fragments. A LISP header is then prepended to each fragment. The size of the encapsulated fragments is then $(S/2 + H)$, which is less than the ITR's estimate of the path MTU between the ITR and its correspondent ETR.

When an ETR receives encapsulated fragments, it treats them as two individually encapsulated packets. It strips the LISP headers and then forwards each fragment to the destination host of the destination site. The two fragments are reassembled at the destination host into the single IP datagram that was originated by the source host. Note that reassembly can happen at the ETR if the encapsulated packet was fragmented at or after the ITR.

This behavior MUST be performed by the ITR only when the source host originates a packet with the 'DF' field of the IP header set to 0. When the 'DF' field of the IP header is set to 1, or the packet is an IPv6 packet originated by the source host, the ITR will drop the packet when the size (adding in the size of the encapsulation header) is greater than L and send an ICMPv4 ICMP Unreachable/Fragmentation-Needed or ICMPv6 "Packet Too Big" message to the source with a value of S, where S is $(L - H)$.

When the outer-header encapsulation uses an IPv4 header, an implementation SHOULD set the DF bit to 1 so ETR fragment reassembly can be avoided. An implementation MAY set the DF bit in such headers to 0 if it has good reason to believe there are unresolvable path MTU issues between the sending ITR and the receiving ETR.

This specification RECOMMENDS that L be defined as 1500. Additional information about in-network MTU and fragmentation issues can be found at [RFC4459].

7.2. A Stateful Solution to MTU Handling

An ITR stateful solution to handle MTU issues is described as follows:

1. The ITR will keep state of the effective MTU for each Locator per Map-Cache entry. The effective MTU is what the core network can deliver along the path between the ITR and ETR.
2. When an IPv4-encapsulated packet with the DF bit set to 1, exceeds what the core network can deliver, one of the intermediate routers on the path will send an an ICMPv4 Unreachable/Fragmentation-Needed to the ITR, respectively. The ITR will parse the ICMP message to determine which Locator is affected by the effective MTU change and then record the new effective MTU value in the Map-Cache entry.
3. When a packet is received by the ITR from a source inside of the site and the size of the packet is greater than the effective MTU stored with the Map-Cache entry associated with the destination EID the packet is for, the ITR will send an ICMPv4 ICMP Unreachable/Fragmentation-Needed message back to the source. The packet size advertised by the ITR in the ICMP message is the effective MTU minus the LISP encapsulation length.

Even though this mechanism is stateful, it has advantages over the stateless IP fragmentation mechanism, by not involving the destination host with reassembly of ITR fragmented packets.

Please note that [RFC1191] and [RFC1981], which describe the use of ICMP packets for PMTU discovery, can behave suboptimally in the presence of ICMP black holes or off-path attackers that spoof ICMP. Possible mitigations include ITRs and ETRs cooperating on MTU probe packets ([RFC4821], [I-D.ietf-tsvwg-datagram-plpmtud]), or ITRs storing the beginning of large packets to verify that they match the echoed packet in ICMP Frag Needed/PTB.

8. Using Virtualization and Segmentation with LISP

There are several cases where segregation is needed at the EID level. For instance, this is the case for deployments containing overlapping addresses, traffic isolation policies or multi-tenant virtualization. For these and other scenarios where segregation is needed, Instance IDs are used.

An Instance ID can be carried in a LISP-encapsulated packet. An ITR that prepends a LISP header will copy a 24-bit value used by the LISP router to uniquely identify the address space. The value is copied to the 'Instance ID' field of the LISP header, and the I-bit is set to 1.

When an ETR decapsulates a packet, the Instance ID from the LISP header is used as a table identifier to locate the forwarding table to use for the inner destination EID lookup.

For example, an 802.1Q VLAN tag or VPN identifier could be used as a 24-bit Instance ID. See [I-D.ietf-lisp-vpn] for LISP VPN use-case details. Please note that the Instance ID is not protected, an on-path attacker can modify the tags and for instance, allow communications between logically isolated VLANs.

Participants within a LISP deployment must agree on the meaning of Instance ID values. The source and destination EIDs MUST belong to the same Instance ID.

Instance ID SHOULD NOT be used with overlapping IPv6 EID addresses.

9. Routing Locator Selection

The Map-Cache contains the state used by ITRs and PITRs to encapsulate packets. When an ITR/PITR receives a packet from inside the LISP site to a destination outside of the site a longest-prefix match lookup of the EID is done to the Map-Cache (see Section 6). The lookup returns a single Locator-Set containing a list of RLOCs corresponding to the EID's topological location. Each RLOC in the Locator-Set is associated with a 'Priority' and 'Weight', this information is used to select the RLOC to encapsulate.

The RLOC with the lowest 'Priority' is selected. An RLOC with 'Priority' 255 means that MUST NOT be used for forwarding. When multiple RLOCs have the same 'Priority' then the 'Weight' states how to load balance traffic among them. The value of the 'Weight' represents the relative weight of the total packets that match the mapping entry.

The following are different scenarios for choosing RLOCs and the controls that are available:

- * The server-side returns one RLOC. The client-side can only use one RLOC. The server-side has complete control of the selection.
- * The server-side returns a list of RLOCs where a subset of the list has the same best Priority. The client can only use the subset list according to the weighting assigned by the server-side. In this case, the server-side controls both the subset list and load-splitting across its members. The client-side can use RLOCs outside of the subset list if it determines that the subset list is unreachable (unless RLOCs are set to a Priority of 255). Some sharing of control exists: the server-side determines the destination RLOC list and load distribution while the client-side has the option of using alternatives to this list if RLOCs in the list are unreachable.
- * The server-side sets a Weight of zero for the RLOC subset list. In this case, the client-side can choose how the traffic load is spread across the subset list. See Section 12 for details on load-sharing mechanisms. Control is shared by the server-side determining the list and the client-side determining load distribution. Again, the client can use alternative RLOCs if the server-provided list of RLOCs is unreachable.
- * Either side (more likely the server-side ETR) decides to "glean" the RLOCs. For example, if the server-side ETR gleans RLOCs, then the client-side ITR gives the client-side ITR responsibility for bidirectional RLOC reachability and preferability. Server-side ETR gleaning of the client-side ITR RLOC is done by caching the inner-header source EID and the outer-header source RLOC of received packets. The client-side ITR controls how traffic is returned and can alternate using an outer-header source RLOC, which then can be added to the list the server-side ETR uses to return traffic. Since no Priority or Weights are provided using this method, the server-side ETR MUST assume that each client-side ITR RLOC uses the same best Priority with a Weight of zero. In addition, since EID-Prefix encoding cannot be conveyed in data packets, the EID-to-RLOC Cache on Tunnel Routers can grow to be very large. Gleaning has several important considerations. A "gleaned" Map-Cache entry is only stored and used for a RECOMMENDED period of 3 seconds, pending verification. Verification MUST be performed by sending a Map-Request to the source EID (the inner-header IP source address) of the received encapsulated packet. A reply to this "verifying Map-Request" is used to fully populate the Map-Cache entry for the "gleaned" EID and is stored and used for the time indicated from the 'TTL' field

of a received Map-Reply. When a verified Map- Cache entry is stored, data gleaning no longer occurs for subsequent packets that have a source EID that matches the EID-Prefix of the verified entry. This "gleaning" mechanism MUST NOT be used over the public Internet and SHOULD only be used in trusted and closed deployments. Refer to Section 16 for security issues regarding this mechanism.

RLOCs that appear in EID-to-RLOC Map-Reply messages are assumed to be reachable when the R-bit [I-D.ietf-lisp-rfc6833bis] for the Locator record is set to 1. When the R-bit is set to 0, an ITR or PITR MUST NOT encapsulate to the RLOC. Neither the information contained in a Map-Reply nor that stored in the mapping database system provides reachability information for RLOCs. Note that reachability is not part of the mapping system and is determined using one or more of the Routing Locator reachability algorithms described in the next section.

10. Routing Locator Reachability

Several Data-Plane mechanisms for determining RLOC reachability are currently defined. Please note that additional Control-Plane based reachability mechanisms are defined in [I-D.ietf-lisp-rfc6833bis].

1. An ETR MAY examine the Locator-Status-Bits in the LISP header of an encapsulated data packet received from an ITR. If the ETR is also acting as an ITR and has traffic to return to the original ITR site, it can use this status information to help select an RLOC.
2. When an ETR receives an encapsulated packet from an ITR, the source RLOC from the outer header of the packet is likely to be reachable. Please note that in some scenarios the RLOC from the outer header can be an spoofable field.
3. An ITR/ETR pair can use the 'Echo-Noncing' Locator reachability algorithms described in this section.

When determining Locator up/down reachability by examining the Locator-Status-Bits from the LISP-encapsulated data packet, an ETR will receive up-to-date status from an encapsulating ITR about reachability for all ETRs at the site. CE-based ITRs at the source site can determine reachability relative to each other using the site IGP as follows:

- * Under normal circumstances, each ITR will advertise a default route into the site IGP.

- * If an ITR fails or if the upstream link to its PE fails, its default route will either time out or be withdrawn.

Each ITR can thus observe the presence or lack of a default route originated by the others to determine the Locator-Status-Bits it sets for them.

When ITRs at the site are not deployed in CE routers, the IGP can still be used to determine the reachability of Locators, provided they are injected into the IGP. This is typically done when a /32 address is configured on a loopback interface.

RLOCs listed in a Map-Reply are numbered with ordinals 0 to n-1. The Locator-Status-Bits in a LISP-encapsulated packet are numbered from 0 to n-1 starting with the least significant bit. For example, if an RLOC listed in the 3rd position of the Map-Reply goes down (ordinal value 2), then all ITRs at the site will clear the 3rd least significant bit (xxxx x0xx) of the 'Locator-Status-Bits' field for the packets they encapsulate.

When an xTR decides to use 'Locator-Status-Bits' to affect reachability information, it acts as follows: ETRs decapsulating a packet will check for any change in the 'Locator-Status-Bits' field. When a bit goes from 1 to 0, the ETR, if acting also as an ITR, will refrain from encapsulating packets to an RLOC that is indicated as down. It will only resume using that RLOC if the corresponding Locator-Status-Bit returns to a value of 1. Locator-Status-Bits are associated with a Locator-Set per EID-Prefix. Therefore, when a Locator becomes unreachable, the Locator-Status-Bit that corresponds to that Locator's position in the list returned by the last Map-Reply will be set to zero for that particular EID-Prefix.

Locator-Status-Bits MUST NOT be used over the public Internet and SHOULD only be used in trusted and closed deployments. In addition Locator-Status-Bits SHOULD be coupled with Map-Versioning [I-D.ietf-lisp-6834bis] to prevent race conditions where Locator-Status-Bits are interpreted as referring to different RLOCs than intended. Refer to Section 16 for security issues regarding this mechanism.

If an ITR encapsulates a packet to an ETR and the packet is received and decapsulated by the ETR, it is implied but not confirmed by the ITR that the ETR's RLOC is reachable. In most cases, the ETR can also reach the ITR but cannot assume this to be true, due to the possibility of path asymmetry. In the presence of unidirectional traffic flow from an ITR to an ETR, the ITR SHOULD NOT use the lack of return traffic as an indication that the ETR is unreachable. Instead, it MUST use an alternate mechanism to determine reachability.

The security considerations of Section 16 related to data-plane reachability applies to the data-plane RLOC reachability mechanisms described in this section.

10.1. Echo Nonce Algorithm

When data flows bidirectionally between Locators from different sites, a Data-Plane mechanism called "nonce echoing" can be used to determine reachability between an ITR and ETR. When an ITR wants to solicit a nonce echo, it sets the N- and E-bits and places a 24-bit nonce [RFC4086] in the LISP header of the next encapsulated data packet.

When this packet is received by the ETR, the encapsulated packet is forwarded as normal. When the ETR is an xTR (co-located as an ITR), it then sends a data packet to the ITR (when it is an xTR co-located as an ETR), it includes the nonce received earlier with the N-bit set and E-bit cleared. The ITR sees this "echoed nonce" and knows that the path to and from the ETR is up.

The ITR will set the E-bit and N-bit for every packet it sends while in the echo-nonce-request state. The time the ITR waits to process the echoed nonce before it determines the path is unreachable is variable and is a choice left for the implementation.

If the ITR is receiving packets from the ETR but does not see the nonce echoed while being in the echo-nonce-request state, then the path to the ETR is unreachable. This decision MAY be overridden by other Locator reachability algorithms. Once the ITR determines that the path to the ETR is down, it can switch to another Locator for that EID-Prefix.

Note that "ITR" and "ETR" are relative terms here. Both devices MUST be implementing both ITR and ETR functionality for the echo nonce mechanism to operate.

The ITR and ETR MAY both go into the echo-nonce-request state at the same time. The number of packets sent or the time during which echo nonce requests are sent is an implementation-specific setting. In this case, an xTR receiving the echo-nonce-request packets will suspend the echo-nonce-request state and setup a 'echo-nonce-request-state' timer. After the 'echo-nonce-request-state' timer expires it will resume the echo-nonce-request state.

This mechanism does not completely solve the forward path reachability problem, as traffic may be unidirectional. That is, the ETR receiving traffic at a site MAY not be the same device as an ITR that transmits traffic from that site, or the site-to-site traffic is unidirectional so there is no ITR returning traffic.

The echo-nonce algorithm is bilateral. That is, if one side sets the E-bit and the other side is not enabled for echo-nonce, then the echoing of the nonce does not occur and the requesting side may erroneously consider the Locator unreachable. An ITR SHOULD set the E-bit in an encapsulated data packet when it knows the ETR is enabled for echo-nonce. This is conveyed by the E-bit in the Map-Reply message.

Many implementations default to not advertising they are echo-nonce capable in Map-Reply messages and so RLOC-probing tends to be used for RLOC reachability.

The echo-nonce mechanism MUST NOT be used over the public Internet and MUST only be used in trusted and closed deployments. Refer to Section 16 for security issues regarding this mechanism.

11. EID Reachability within a LISP Site

A site MAY be multihomed using two or more ETRs. The hosts and infrastructure within a site will be addressed using one or more EID-Prefixes that are mapped to the RLOCs of the relevant ETRs in the mapping system. One possible failure mode is for an ETR to lose reachability to one or more of the EID-Prefixes within its own site. When this occurs when the ETR sends Map-Replies, it can clear the R-bit associated with its own Locator. And when the ETR is also an ITR, it can clear its Locator-Status-Bit in the encapsulation data header.

It is recognized that there are no simple solutions to the site partitioning problem because it is hard to know which part of the EID-Prefix range is partitioned and which Locators can reach any sub-ranges of the EID-Prefixes. Note that this is not a new problem introduced by the LISP architecture. The problem exists today when a multihomed site uses BGP to advertise its reachability upstream.

12. Routing Locator Hashing

When an ETR provides an EID-to-RLOC mapping in a Map-Reply message that is stored in the Map-Cache of a requesting ITR, the Locator-Set for the EID-Prefix MAY contain different Priority and Weight values for each locator address. When more than one best Priority Locator exists, the ITR can decide how to load-share traffic against the corresponding Locators.

The following hash algorithm MAY be used by an ITR to select a Locator for a packet destined to an EID for the EID-to-RLOC mapping:

1. Either a source and destination address hash or the traditional 5-tuple hash can be used. The traditional 5-tuple hash includes the source and destination addresses; source and destination TCP, UDP, or Stream Control Transmission Protocol (SCTP) port numbers; and the IP protocol number field or IPv6 next-protocol fields of a packet that a host originates from within a LISP site. When a packet is not a TCP, UDP, or SCTP packet, the source and destination addresses only from the header are used to compute the hash.
2. Take the hash value and divide it by the number of Locators stored in the Locator-Set for the EID-to-RLOC mapping.
3. The remainder will yield a value of 0 to "number of Locators minus 1". Use the remainder to select the Locator in the Locator-Set.

The specific hash algorithm the ITR uses for load-sharing is out of scope for this document and does not prevent interoperability.

The Source port SHOULD be the same for all packets belonging to the same flow. Also note that when a packet is LISP encapsulated, the source port number in the outer UDP header needs to be set. Selecting a hashed value allows core routers that are attached to Link Aggregation Groups (LAGs) to load-split the encapsulated packets across member links of such LAGs. Otherwise, core routers would see a single flow, since packets have a source address of the ITR, for packets that are originated by different EIDs at the source site. A suggested setting for the source port number computed by an ITR is a 5-tuple hash function on the inner header, as described above. The source port SHOULD be the same for all packets belonging to the same flow.

Many core router implementations use a 5-tuple hash to decide how to balance packet load across members of a LAG. The 5-tuple hash includes the source and destination addresses of the packet and the

source and destination ports when the protocol number in the packet is TCP or UDP. For this reason, UDP encoding is used for LISP encapsulation. In this scenario, when the outer header is IPv6, the flow label MAY also be set following the procedures specified in [RFC6438]. When the inner header is IPv6 then the flow label is not zero, it MAY be used to compute the hash.

13. Changing the Contents of EID-to-RLOC Mappings

Since the LISP architecture uses a caching scheme to retrieve and store EID-to-RLOC mappings, the only way an ITR can get a more up-to-date mapping is to re-request the mapping. However, the ITRs do not know when the mappings change, and the ETRs do not keep track of which ITRs requested its mappings. For scalability reasons, it is desirable to maintain this approach but need to provide a way for ETRs to change their mappings and inform the sites that are currently communicating with the ETR site using such mappings.

This section defines two Data-Plane mechanism for updating EID-to-RLOC mappings. Additionally, the Solicit-Map Request (SMR) Control-Plane updating mechanism is specified in [I-D.ietf-lisp-rfc6833bis].

13.1. Locator-Status-Bits

Locator-Status-Bits (LSB) can also be used to keep track of the Locator status (up or down) when EID-to-RLOC mappings are changing. When LSB are used in a LISP deployment, all LISP tunnel routers MUST implement both ITR and ETR capabilities (therefore all tunnel routers are effectively xTRs). In this section the term "source xTR" is used to refer to the xTR setting the LSB and "destination xTR" is used to refer to the xTR receiving the LSB. The procedure is as follows:

First, when a Locator record is added or removed from the Locator-Set, the source xTR will signal this by sending a Solicit-Map Request (SMR) Control-Plane message [I-D.ietf-lisp-rfc6833bis] to the destination xTR. At this point the source xTR MUST NOT use LSB (L-bit = 0) since the destination xTR site has outdated information. The source xTR will setup a 'use-LSB' timer.

Second and as defined in [I-D.ietf-lisp-rfc6833bis], upon reception of the SMR message the destination xTR will retrieve the updated EID-to-RLOC mappings by sending a Map-Request.

And third, when the 'use-LSB' timer expires, the source xTR can use again LSB with the destination xTR to signal the Locator status (up or down). The specific value for the 'use-LSB' timer depends on the LISP deployment, the 'use-LSB' timer needs to be large enough for the destination xTR to retrieve the updated EID-to-RLOC mappings. A RECOMMENDED value for the 'use-LSB' timer is 5 minutes.

13.2. Database Map-Versioning

When there is unidirectional packet flow between an ITR and ETR, and the EID-to-RLOC mappings change on the ETR, it needs to inform the ITR so encapsulation to a removed Locator can stop and can instead be started to a new Locator in the Locator-Set.

An ETR, can send Map-Reply messages carrying a Map-Version Number in an EID-record. This is known as the Destination Map-Version Number. ITRs include the Destination Map-Version Number in packets they encapsulate to the site.

An ITR, when it encapsulates packets to ETRs, can convey its own Map-Version Number. This is known as the Source Map-Version Number.

When presented in EID-records of Map-Register messages, a Map-Version Number is a good way for the Map-Server to assure that all ETRs for a site registering to it are synchronized according to Map-Version Number.

See [I-D.ietf-lisp-6834bis] for a more detailed analysis and description of Database Map-Versioning.

14. Multicast Considerations

A multicast group address, as defined in the original Internet architecture, is an identifier of a grouping of topologically independent receiver host locations. The address encoding itself does not determine the location of the receiver(s). The multicast routing protocol, and the network-based state the protocol creates, determine where the receivers are located.

In the context of LISP, a multicast group address is both an EID and a Routing Locator. Therefore, no specific semantic or action needs to be taken for a destination address, as it would appear in an IP header. Therefore, a group address that appears in an inner IP header built by a source host will be used as the destination EID. The outer IP header (the destination Routing Locator address), prepended by a LISP router, can use the same group address as the destination Routing Locator, use a multicast or unicast Routing Locator obtained from a Mapping System lookup, or use other means to determine the group address mapping.

With respect to the source Routing Locator address, the ITR prepends its own IP address as the source address of the outer IP header, just like it would if the destination EID was a unicast address. This source Routing Locator address, like any other Routing Locator address, MUST be routable on the underlay.

There are two approaches for LISP-Multicast, one that uses native multicast routing in the underlay with no support from the Mapping System and the other that uses only unicast routing in the underlay with support from the Mapping System. See [RFC6831] and [RFC8378], respectively, for details. Details for LISP-Multicast and interworking with non-LISP sites are described in [RFC6831] and [RFC6832].

15. Router Performance Considerations

LISP is designed to be very "hardware-based forwarding friendly". A few implementation techniques can be used to incrementally implement LISP:

- * When a tunnel-encapsulated packet is received by an ETR, the outer destination address may not be the address of the router. This makes it challenging for the control plane to get packets from the hardware. This may be mitigated by creating special Forwarding Information Base (FIB) entries for the EID-Prefixes of EIDs served by the ETR (those for which the router provides an RLOC translation). These FIB entries are marked with a flag indicating that Control-Plane processing SHOULD be performed. The forwarding logic of testing for particular IP protocol number values is not necessary. There are a few proven cases where no changes to existing deployed hardware were needed to support the LISP Data-Plane.

- * On an ITR, prepending a new IP header consists of adding more octets to a MAC rewrite string and prepending the string as part of the outgoing encapsulation procedure. Routers that support Generic Routing Encapsulation (GRE) tunneling [RFC2784] or 6to4 tunneling [RFC3056] may already support this action.
- * A packet's source address or interface the packet was received on can be used to select VRF (Virtual Routing/Forwarding). The VRF's routing table can be used to find EID-to-RLOC mappings.

For performance issues related to Map-Cache management, see Section 16.

16. Security Considerations

In what follows we highlight security considerations that apply when LISP is deployed in environments such as those specified in Section 1.1.

The optional mechanisms of gleaning is offered to directly obtain a mapping from the LISP encapsulated packets. Specifically, an xTR can learn the EID-to-RLOC mapping by inspecting the source RLOC and source EID of an encapsulated packet, and insert this new mapping into its Map-Cache. An off-path attacker can spoof the source EID address to divert the traffic sent to the victim's spoofed EID. If the attacker spoofs the source RLOC, it can mount a DoS attack by redirecting traffic to the spoofed victim's RLOC, potentially overloading it.

The LISP Data-Plane defines several mechanisms to monitor RLOC Data-Plane reachability, in this context Locator-Status Bits, Nonce-Present and Echo-Nonce bits of the LISP encapsulation header can be manipulated by an attacker to mount a DoS attack. An off-path attacker able to spoof the RLOC and/or nonce of a victim's xTR can manipulate such mechanisms to declare false information about the RLOC's reachability status.

For example of such attacks, an off-path attacker can exploit the echo-nonce mechanism by sending data packets to an ITR with a random nonce from an ETR's spoofed RLOC. Note the attacker must guess a valid nonce the ITR is requesting to be echoed within a small window of time. The goal is to convince the ITR that the ETR's RLOC is reachable even when it may not be reachable. If the attack is successful, the ITR believes the wrong reachability status of the ETR's RLOC until RLOC-probing detects the correct status. This time frame is on the order of 10s of seconds. This specific attack can be mitigated by preventing RLOC spoofing in the network by deploying uRPF BCP 38 [RFC2827]. In addition and in order to exploit this

vulnerability, the off-path attacker must send echo-nonce packets at high rate. If the nonces have never been requested by the ITR, it can protect itself from erroneous reachability attacks.

A LISP-specific uRPF check is also possible. When decapsulating, an ETR can check that the source EID and RLOC are valid EID-to-RLOC mappings by checking the Mapping System.

Map-Versioning is a Data-Plane mechanism used to signal a peering xTR that a local EID-to-RLOC mapping has been updated, so that the peering xTR uses LISP Control-Plane signaling message to retrieve a fresh mapping. This can be used by an attacker to forge the map-versioning field of a LISP encapsulated header and force an excessive amount of signaling between xTRs that may overload them. Further security considerations on Map-Versioning can be found in [I-D.ietf-lisp-6834bis].

Locator-Status-Bits, echo-nonce and map-versioning MUST NOT be used over the public Internet and SHOULD only be used in trusted and closed deployments. In addition Locator-Status-Bits SHOULD be coupled with map-versioning to prevent race conditions where Locator-Status-Bits are interpreted as referring to different RLOCs than intended.

LISP implementations and deployments which permit outer header fragments of IPv6 LISP encapsulated packets as a means of dealing with MTU issues should also use implementation techniques in ETRs to prevent this from being a DoS attack vector. Limits on the number of fragments awaiting reassembly at an ETR, RTR, or PETR, and the rate of admitting such fragments may be used.

17. Network Management Considerations

Considerations for network management tools exist so the LISP protocol suite can be operationally managed. These mechanisms can be found in [RFC7052] and [RFC6835].

18. Changes since RFC 6830

For implementation considerations, the following changes have been made to this document since RFC 6830 was published:

- * It is no longer mandated that a maximum number of 2 LISP headers be prepended to a packet. If there is a application need for more than 2 LISP headers, an implementation can support more. However, it is RECOMMENDED that a maximum of two LISP headers can be prepended to a packet.

- * The 3 reserved flag bits in the LISP header have been allocated for [RFC8061]. The low-order 2 bits of the 3-bit field (now named the KK bits) are used as a key identifier. The 1 remaining bit is still documented as reserved and unassigned.
- * Data-Plane gleaning for creating map-cache entries has been made optional. Any ITR implementations that depend on or assume the remote ETR is gleaning should not do so. This does not create any interoperability problems since the control-plane map-cache population procedures are unilateral and are the typical method for map-cache population.
- * The bulk of the changes to this document which reduces its length are due to moving the LISP control-plane messaging and procedures to [I-D.ietf-lisp-rfc6833bis].

19. IANA Considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to this Data-Plane LISP specification, in accordance with BCP 26 [RFC8126].

19.1. LISP UDP Port Numbers

The IANA registry has allocated UDP port number 4341 for the LISP Data-Plane. IANA has updated the description for UDP port 4341 as follows:

lisp-data	4341 udp	LISP Data Packets
-----------	----------	-------------------

20. References

20.1. Normative References

[I-D.ietf-lisp-6834bis]
Iannone, L., Saucez, D., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Map-Versioning", Work in Progress, Internet-Draft, draft-ietf-lisp-6834bis-10, 3 May 2022, <<https://www.ietf.org/archive/id/draft-ietf-lisp-6834bis-10.txt>>.

[I-D.ietf-lisp-rfc6833bis]
Farinacci, D., Maino, F., Fuller, V., and A. Cabellos, "Locator/ID Separation Protocol (LISP) Control-Plane", Work in Progress, Internet-Draft, draft-ietf-lisp-rfc6833bis-31, 2 May 2022, <<https://www.ietf.org/archive/id/draft-ietf-lisp-rfc6833bis-31.txt>>.

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <<https://www.rfc-editor.org/info/rfc768>>.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <<https://www.rfc-editor.org/info/rfc2474>>.
- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <<https://www.rfc-editor.org/info/rfc2827>>.
- [RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <<https://www.rfc-editor.org/info/rfc2983>>.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<https://www.rfc-editor.org/info/rfc6040>>.
- [RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <<https://www.rfc-editor.org/info/rfc6438>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC6831] Farinacci, D., Meyer, D., Zwiebel, J., and S. Venaas, "The Locator/ID Separation Protocol (LISP) for Multicast Environments", RFC 6831, DOI 10.17487/RFC6831, January 2013, <<https://www.rfc-editor.org/info/rfc6831>>.

- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8378] Moreno, V. and D. Farinacci, "Signal-Free Locator/ID Separation Protocol (LISP) Multicast", RFC 8378, DOI 10.17487/RFC8378, May 2018, <<https://www.rfc-editor.org/info/rfc8378>>.

20.2. Informative References

- [AFN] IANA, "Address Family Numbers", August 2016, <<http://www.iana.org/assignments/address-family-numbers>>.
- [CHIAPPA] Chiappa, J., "Endpoints and Endpoint names: A Proposed", 1999, <<http://mercury.lcs.mit.edu/~jnc/tech/endpoints.txt>>.
- [I-D.ietf-lisp-introduction] Cabellos, A. and D. S. (Ed.), "An Architectural Introduction to the Locator/ID Separation Protocol (LISP)", Work in Progress, Internet-Draft, draft-ietf-lisp-introduction-15, 20 September 2021, <<https://www.ietf.org/archive/id/draft-ietf-lisp-introduction-15.txt>>.
- [I-D.ietf-lisp-vpn] Moreno, V. and D. Farinacci, "LISP Virtual Private Networks (VPNs)", Work in Progress, Internet-Draft, draft-ietf-lisp-vpn-08, 18 January 2022, <<https://www.ietf.org/archive/id/draft-ietf-lisp-vpn-08.txt>>.

- [I-D.ietf-tsvwg-datagram-plpmtud]
Fairhurst, G., Jones, T., Tuexen, M., Ruengeler, I., and
T. Voelker, "Packetization Layer Path MTU Discovery for
Datagram Transports", Work in Progress, Internet-Draft,
draft-ietf-tsvwg-datagram-plpmtud-22, 10 June 2020,
<[https://www.ietf.org/archive/id/draft-ietf-tsvwg-
datagram-plpmtud-22.txt](https://www.ietf.org/archive/id/draft-ietf-tsvwg-datagram-plpmtud-22.txt)>.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities",
STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987,
<<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191,
DOI 10.17487/RFC1191, November 1990,
<<https://www.rfc-editor.org/info/rfc1191>>.
- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G.
J., and E. Lear, "Address Allocation for Private
Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918,
February 1996, <<https://www.rfc-editor.org/info/rfc1918>>.
- [RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery
for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August
1996, <<https://www.rfc-editor.org/info/rfc1981>>.
- [RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P.
Traina, "Generic Routing Encapsulation (GRE)", RFC 2784,
DOI 10.17487/RFC2784, March 2000,
<<https://www.rfc-editor.org/info/rfc2784>>.
- [RFC3056] Carpenter, B. and K. Moore, "Connection of IPv6 Domains
via IPv4 Clouds", RFC 3056, DOI 10.17487/RFC3056, February
2001, <<https://www.rfc-editor.org/info/rfc3056>>.
- [RFC3232] Reynolds, J., Ed., "Assigned Numbers: RFC 1700 is Replaced
by an On-line Database", RFC 3232, DOI 10.17487/RFC3232,
January 2002, <<https://www.rfc-editor.org/info/rfc3232>>.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
A., Peterson, J., Sparks, R., Handley, M., and E.
Schooler, "SIP: Session Initiation Protocol", RFC 3261,
DOI 10.17487/RFC3261, June 2002,
<<https://www.rfc-editor.org/info/rfc3261>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker,
"Randomness Requirements for Security", BCP 106, RFC 4086,
DOI 10.17487/RFC4086, June 2005,
<<https://www.rfc-editor.org/info/rfc4086>>.

- [RFC4459] Savola, P., "MTU and Fragmentation Issues with In-the-Network Tunneling", RFC 4459, DOI 10.17487/RFC4459, April 2006, <<https://www.rfc-editor.org/info/rfc4459>>.
- [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <<https://www.rfc-editor.org/info/rfc4821>>.
- [RFC4984] Meyer, D., Ed., Zhang, L., Ed., and K. Fall, Ed., "Report from the IAB Workshop on Routing and Addressing", RFC 4984, DOI 10.17487/RFC4984, September 2007, <<https://www.rfc-editor.org/info/rfc4984>>.
- [RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <<https://www.rfc-editor.org/info/rfc6832>>.
- [RFC6835] Farinacci, D. and D. Meyer, "The Locator/ID Separation Protocol Internet Groper (LIG)", RFC 6835, DOI 10.17487/RFC6835, January 2013, <<https://www.rfc-editor.org/info/rfc6835>>.
- [RFC6935] Eubanks, M., Chimento, P., and M. Westerlund, "IPv6 and UDP Checksums for Tunneled Packets", RFC 6935, DOI 10.17487/RFC6935, April 2013, <<https://www.rfc-editor.org/info/rfc6935>>.
- [RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <<https://www.rfc-editor.org/info/rfc6936>>.
- [RFC7052] Schudel, G., Jain, A., and V. Moreno, "Locator/ID Separation Protocol (LISP) MIB", RFC 7052, DOI 10.17487/RFC7052, October 2013, <<https://www.rfc-editor.org/info/rfc7052>>.
- [RFC7215] Jakab, L., Cabellos-Aparicio, A., Coras, F., Domingo-Pascual, J., and D. Lewis, "Locator/Identifier Separation Protocol (LISP) Network Element Deployment Considerations", RFC 7215, DOI 10.17487/RFC7215, April 2014, <<https://www.rfc-editor.org/info/rfc7215>>.
- [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.

- [RFC8061] Farinacci, D. and B. Weis, "Locator/ID Separation Protocol (LISP) Data-Plane Confidentiality", RFC 8061, DOI 10.17487/RFC8061, February 2017, <<https://www.rfc-editor.org/info/rfc8061>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.

Appendix A. Acknowledgments

An initial thank you goes to Dave Oran for planting the seeds for the initial ideas for LISP. His consultation continues to provide value to the LISP authors.

A special and appreciative thank you goes to Noel Chiappa for providing architectural impetus over the past decades on separation of location and identity, as well as detailed reviews of the LISP architecture and documents, coupled with enthusiasm for making LISP a practical and incremental transition for the Internet.

The original authors would like to gratefully acknowledge many people who have contributed discussions and ideas to the making of this proposal. They include Scott Brim, Andrew Partan, John Zwiebel, Jason Schiller, Lixia Zhang, Dorian Kim, Peter Schoenmaker, Vijay Gill, Geoff Huston, David Conrad, Mark Handley, Ron Bonica, Ted Seely, Mark Townsley, Chris Morrow, Brian Weis, Dave McGrew, Peter Lothberg, Dave Thaler, Eliot Lear, Shane Amante, Ved Kafle, Olivier Bonaventure, Luigi Iannone, Robin Whittle, Brian Carpenter, Joel Halpern, Terry Manderson, Roger Jorgensen, Ran Atkinson, Stig Venaas, Iljitsch van Beijnum, Roland Bless, Dana Blair, Bill Lynch, Marc Woolward, Damien Saucez, Damian Lezama, Attila De Groot, Parantap Lahiri, David Black, Roque Gagliano, Isidor Kouvelas, Jesper Skriver, Fred Templin, Margaret Wasserman, Sam Hartman, Michael Hofling, Pedro Marques, Jari Arkko, Gregg Schudel, Srinivas Subramanian, Amit Jain, Xu Xiaohu, Dharendra Trivedi, Yakov Rekhter, John Scudder, John Drake, Dimitri Papadimitriou, Ross Callon, Selina Heimlich, Job Snijders, Vina Ermagan, Fabio Maino, Victor Moreno, Chris White, Clarence Filsfils, Alia Atlas, Florin Coras and Alberto Rodriguez.

This work originated in the Routing Research Group (RRG) of the IRTF. An individual submission was converted into the IETF LISP working group document that became this RFC.

The LISP working group would like to give a special thanks to Jari Arkko, the Internet Area AD at the time that the set of LISP documents were being prepared for IESG last call, and for his meticulous reviews and detailed commentaries on the 7 working group last call documents progressing toward standards-track RFCs.

The current authors would like to give a sincere thank you to the people who help put LISP on standards track in the IETF. They include Joel Halpern, Luigi Iannone, Deborah Brungard, Fabio Maino, Scott Bradner, Kyle Rose, Takeshi Takahashi, Sarah Banks, Pete Resnick, Colin Perkins, Mirja Kuhlewind, Francis Dupont, Benjamin Kaduk, Eric Rescorla, Alvaro Retana, Alexey Melnikov, Alissa Cooper, Suresh Krishnan, Alberto Rodriguez-Natal, Vina Ermagen, Mohamed Boucadair, Brian Trammell, Sabrina Tanamal, and John Drake. The contributions they offered greatly added to the security, scale, and robustness of the LISP architecture and protocols.

Appendix B. Document Change Log

[RFC Editor: Please delete this section on publication as RFC.]

B.1. Changes to draft-ietf-lisp-rfc6830bis-38

- * Posted May 2022.
- * Removed detailed paragraphs in Section 13.2 that is duplicated text from [I-D.ietf-lisp-6834bis], which is the authoritative source for Map Versioning.

B.2. Changes to draft-ietf-lisp-rfc6830bis-37

- * Posted May 2022.
- * Added references to 6834bis instead of pointing text to Section 13.2. This is so we can advance the Map-Versioning draft rfc6834bis to proposed standard.

B.3. Changes to draft-ietf-lisp-rfc6830bis-28

- * Posted November 2019.
- * Fixed how LSB behave in the presence of new/removed locators.
- * Added ETR synchronization requirements when using Map-Versioning.
- * Fixed a large set of minor comments and edits.

B.4. Changes to draft-ietf-lisp-rfc6830bis-27

- * Posted April 2019 post telechat.
- * Made editorial corrections per Warren's suggestions.
- * Put in suggested text from Luigi that Mirja agreed with.
- * LSB, Echo-Nonce and Map-Versioning SHOULD be only used in closed environments.
- * Removed paragraph stating that Instance-ID can be 32-bit in the control-plane.
- * 6831/8378 are now normative.
- * Rewritten Security Considerations according to the changes.
- * Stated that LSB SHOULD be coupled with Map-Versioning.

B.5. Changes to draft-ietf-lisp-rfc6830bis-26

- * Posted late October 2018.
- * Changed description about "reserved" bits to state "reserved and unassigned".

B.6. Changes to draft-ietf-lisp-rfc6830bis-25

- * Posted mid October 2018.
- * Added more to the Security Considerations section with discussion about echo-nonce attacks.

B.7. Changes to draft-ietf-lisp-rfc6830bis-24

- * Posted mid October 2018.
- * Final editorial changes for Eric and Ben.

B.8. Changes to draft-ietf-lisp-rfc6830bis-23

- * Posted early October 2018.
- * Added an applicability statement in section 1 to address security concerns from Telechat.

- B.9. Changes to draft-ietf-lisp-rfc6830bis-22
- * Posted early October 2018.
 - * Changes to reflect comments post Telechat.
- B.10. Changes to draft-ietf-lisp-rfc6830bis-21
- * Posted late-September 2018.
 - * Changes to reflect comments from Sep 27th Telechat.
- B.11. Changes to draft-ietf-lisp-rfc6830bis-20
- * Posted late-September 2018.
 - * Fix old reference to RFC3168, changed to RFC6040.
- B.12. Changes to draft-ietf-lisp-rfc6830bis-19
- * Posted late-September 2018.
 - * More editorial changes.
- B.13. Changes to draft-ietf-lisp-rfc6830bis-18
- * Posted mid-September 2018.
 - * Changes to reflect comments from Secdir review (Mirja).
- B.14. Changes to draft-ietf-lisp-rfc6830bis-17
- * Posted September 2018.
 - * Indicate in the "Changes since RFC 6830" section why the document has been shortened in length.
 - * Make reference to RFC 8085 about UDP congestion control.
 - * More editorial changes from multiple IESG reviews.
- B.15. Changes to draft-ietf-lisp-rfc6830bis-16
- * Posted late August 2018.
 - * Distinguish the message type names between ICMP for IPv4 and ICMP for IPv6 for handling MTU issues.

B.16. Changes to draft-ietf-lisp-rfc6830bis-15

- * Posted August 2018.
- * Final editorial changes before RFC submission for Proposed Standard.
- * Added section "Changes since RFC 6830" so implementers are informed of any changes since the last RFC publication.

B.17. Changes to draft-ietf-lisp-rfc6830bis-14

- * Posted July 2018 IETF week.
- * Put obsolete of RFC 6830 in Intro section in addition to abstract.

B.18. Changes to draft-ietf-lisp-rfc6830bis-13

- * Posted March IETF Week 2018.
- * Clarified that a new nonce is required per RLOC.
- * Removed 'Clock Sweep' section. This text must be placed in a new OAM document.
- * Some references changed from normative to informative

B.19. Changes to draft-ietf-lisp-rfc6830bis-12

- * Posted July 2018.
- * Fixed Luigi editorial comments to ready draft for RFC status.

B.20. Changes to draft-ietf-lisp-rfc6830bis-11

- * Posted March 2018.
- * Removed sections 16, 17 and 18 (Mobility, Deployment and Traceroute considerations). This text must be placed in a new OAM document.

B.21. Changes to draft-ietf-lisp-rfc6830bis-10

- * Posted March 2018.
- * Updated section 'Router Locator Selection' stating that the Data-Plane MUST follow what's stored in the Map-Cache (priorities and weights).

- * Section 'Routing Locator Reachability': Removed bullet point 2 (ICMP Network/Host Unreachable), 3 (hints from BGP), 4 (ICMP Port Unreachable), 5 (receive a Map-Reply as a response) and RLOC probing
- * Removed 'Solicit-Map Request'.

B.22. Changes to draft-ietf-lisp-rfc6830bis-09

- * Posted January 2018.
- * Add more details in section 5.3 about DSCP processing during encapsulation and decapsulation.
- * Added clarity to definitions in the Definition of Terms section from various commenters.
- * Removed PA and PI definitions from Definition of Terms section.
- * More editorial changes.
- * Removed 4342 from IANA section and move to RFC6833 IANA section.

B.23. Changes to draft-ietf-lisp-rfc6830bis-08

- * Posted January 2018.
- * Remove references to research work for any protocol mechanisms.
- * Document scanned to make sure it is RFC 2119 compliant.
- * Made changes to reflect comments from document WG shepherd Luigi Iannone.
- * Ran IDNITs on the document.

B.24. Changes to draft-ietf-lisp-rfc6830bis-07

- * Posted November 2017.
- * Rephrase how Instance-IDs are used and don't refer to [RFC1918] addresses.

B.25. Changes to draft-ietf-lisp-rfc6830bis-06

- * Posted October 2017.
- * Put RTR definition before it is used.

- * Rename references that are now working group drafts.
- * Remove "EIDs MUST NOT be used as used by a host to refer to other hosts. Note that EID blocks MAY LISP RLOCs".
- * Indicate what address-family can appear in data packets.
- * ETRs may, rather than will, be the ones to send Map-Replies.
- * Recommend, rather than mandate, max encapsulation headers to 2.
- * Reference VPN draft when introducing Instance-ID.
- * Indicate that SMRs can be sent when ITR/ETR are in the same node.
- * Clarify when private addresses can be used.

B.26. Changes to draft-ietf-lisp-rfc6830bis-05

- * Posted August 2017.
- * Make it clear that a Re-encapsulating Tunnel Router is an RTR.

B.27. Changes to draft-ietf-lisp-rfc6830bis-04

- * Posted July 2017.
- * Changed reference of IPv6 RFC2460 to RFC8200.
- * Indicate that the applicability statement for UDP zero checksums over IPv6 adheres to RFC6936.

B.28. Changes to draft-ietf-lisp-rfc6830bis-03

- * Posted May 2017.
- * Move the control-plane related codepoints in the IANA Considerations section to RFC6833bis.

B.29. Changes to draft-ietf-lisp-rfc6830bis-02

- * Posted April 2017.
- * Reflect some editorial comments from Damien Sausez.

B.30. Changes to draft-ietf-lisp-rfc6830bis-01

- * Posted March 2017.
- * Include references to new RFCs published.
- * Change references from RFC6833 to RFC6833bis.
- * Clarified LCAF text in the IANA section.
- * Remove references to "experimental".

B.31. Changes to draft-ietf-lisp-rfc6830bis-00

- * Posted December 2016.
- * Created working group document from draft-farinacci-lisp-rfc6830-00 individual submission. No other changes made.

Authors' Addresses

Dino Farinacci
lispers.net
Email: farinacci@gmail.com

Vince Fuller
vaf.net Internet Consulting
Email: vince.fuller@gmail.com

Dave Meyer
1-4-5.net
Email: dmm@1-4-5.net

Darrel Lewis
Cisco Systems
170 Tasman Drive
San Jose, CA
United States of America
Email: darlewis@cisco.com

Albert Cabellos
UPC/BarcelonaTech
Campus Nord, C. Jordi Girona 1-3
Barcelona Catalunya
Spain
Email: acabello@ac.upc.edu

Network Working Group
Internet-Draft
Obsoletes: 6830, 6833 (if approved)
Intended status: Standards Track
Expires: November 3, 2022

D. Farinacci
lispers.net
F. Maino
Cisco Systems
V. Fuller
vaf.net Internet Consulting
A. Cabellos (Ed.)
UPC/BarcelonaTech
May 2, 2022

Locator/ID Separation Protocol (LISP) Control-Plane
draft-ietf-lisp-rfc6833bis-31

Abstract

This document describes the Control-Plane and Mapping Service for the Locator/ID Separation Protocol (LISP), implemented by two types of LISP-speaking devices -- the LISP Map-Resolver and LISP Map-Server -- that provides a simplified "front end" for one or more Endpoint ID to Routing Locator mapping databases.

By using this Control-Plane service interface and communicating with Map-Resolvers and Map-Servers, LISP Ingress Tunnel Routers (ITRs) and Egress Tunnel Routers (ETRs) are not dependent on the details of mapping database systems, which facilitates modularity with different database designs. Since these devices implement the "edge" of the LISP Control-Plane infrastructure, connecting EID addressable nodes of a LISP site, it the implementation and operational complexity of the overall cost and effort of deploying LISP.

This document obsoletes RFC 6830 and RFC 6833.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 3, 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Scope of Applicability	5
2. Requirements Notation	5
3. Definition of Terms	5
4. Basic Overview	7
5. LISP IPv4 and IPv6 Control-Plane Packet Formats	8
5.1. LISP Control Packet Type Allocations	11
5.2. Map-Request Message Format	12
5.3. EID-to-RLOC UDP Map-Request Message	14
5.4. Map-Reply Message Format	16
5.5. EID-to-RLOC UDP Map-Reply Message	20
5.6. Map-Register Message Format	23
5.7. Map-Notify/Map-Notify-Ack Message Format	27
5.8. Encapsulated Control Message Format	29
6. Changing the Contents of EID-to-RLOC Mappings	31
6.1. Solicit-Map-Request (SMR)	31
7. Routing Locator Reachability	32
7.1. RLOC-Probing Algorithm	33
8. Interactions with Other LISP Components	34
8.1. ITR EID-to-RLOC Mapping Resolution	34
8.2. EID-Prefix Configuration and ETR Registration	35
8.3. Map-Server Processing	37
8.4. Map-Resolver Processing	37
8.4.1. Anycast Operation	38
9. Security Considerations	38
10. Privacy Considerations	40
11. Changes since RFC 6833	41
12. IANA Considerations	41
12.1. LISP UDP Port Numbers	42

12.2.	LISP Packet Type Codes	42
12.3.	LISP Map-Reply EID-Record Action Codes	42
12.4.	LISP Address Type Codes	43
12.5.	LISP Algorithm ID Numbers	43
12.6.	LISP Bit Flags	44
13.	References	47
13.1.	Normative References	47
13.2.	Informative References	48
Appendix A.	Acknowledgments	53
Appendix B.	Document Change Log	53
B.1.	Changes to draft-ietf-lisp-rfc6833bis-31	53
B.2.	Changes to draft-ietf-lisp-rfc6833bis-26	53
B.3.	Changes to draft-ietf-lisp-rfc6833bis-25	53
B.4.	Changes to draft-ietf-lisp-rfc6833bis-24	54
B.5.	Changes to draft-ietf-lisp-rfc6833bis-23	54
B.6.	Changes to draft-ietf-lisp-rfc6833bis-22	54
B.7.	Changes to draft-ietf-lisp-rfc6833bis-21	54
B.8.	Changes to draft-ietf-lisp-rfc6833bis-20	55
B.9.	Changes to draft-ietf-lisp-rfc6833bis-19	55
B.10.	Changes to draft-ietf-lisp-rfc6833bis-18	55
B.11.	Changes to draft-ietf-lisp-rfc6833bis-17	55
B.12.	Changes to draft-ietf-lisp-rfc6833bis-16	55
B.13.	Changes to draft-ietf-lisp-rfc6833bis-15	56
B.14.	Changes to draft-ietf-lisp-rfc6833bis-14	56
B.15.	Changes to draft-ietf-lisp-rfc6833bis-13	56
B.16.	Changes to draft-ietf-lisp-rfc6833bis-12	56
B.17.	Changes to draft-ietf-lisp-rfc6833bis-11	56
B.18.	Changes to draft-ietf-lisp-rfc6833bis-10	56
B.19.	Changes to draft-ietf-lisp-rfc6833bis-09	57
B.20.	Changes to draft-ietf-lisp-rfc6833bis-08	57
B.21.	Changes to draft-ietf-lisp-rfc6833bis-07	57
B.22.	Changes to draft-ietf-lisp-rfc6833bis-06	58
B.23.	Changes to draft-ietf-lisp-rfc6833bis-05	58
B.24.	Changes to draft-ietf-lisp-rfc6833bis-04	58
B.25.	Changes to draft-ietf-lisp-rfc6833bis-03	58
B.26.	Changes to draft-ietf-lisp-rfc6833bis-02	59
B.27.	Changes to draft-ietf-lisp-rfc6833bis-01	59
B.28.	Changes to draft-ietf-lisp-rfc6833bis-00	59
B.29.	Changes to draft-farinacci-lisp-rfc6833bis-00	59
Authors' Addresses	60

1. Introduction

The Locator/ID Separation Protocol [I-D.ietf-lisp-rfc6830bis] (see also [I-D.ietf-lisp-introduction]) specifies an architecture and mechanism for dynamic tunneling by logically separating the addresses currently used by IP in two separate name spaces: Endpoint IDs (EIDs), used within sites; and Routing Locators (RLOCs), used on the

transit networks that make up the Internet infrastructure. To achieve this separation, LISP defines protocol mechanisms for mapping from EIDs to RLOCs. In addition, LISP assumes the existence of a database to store and propagate those mappings across mapping system nodes. Several such databases have been proposed; among them are the Content distribution Overlay Network Service for LISP-NERD (a Not-so-novel EID-to-RLOC Database) [RFC6837], LISP Alternative Logical Topology (LISP-ALT) [RFC6836], and LISP Delegated Database Tree (LISP-DDT) [RFC8111].

The LISP Mapping Service defines two types of LISP-speaking devices: the Map-Resolver, which accepts Map-Requests from an Ingress Tunnel Router (ITR) and "resolves" the EID-to-RLOC mapping using a mapping database; and the Map-Server, which learns authoritative EID-to-RLOC mappings from an Egress Tunnel Router (ETR) and publishes them in a database.

This LISP Control-Plane Mapping Service can be used by many different encapsulation-based or translation-based Data-Planes which include but are not limited to the ones defined in LISP RFC 6830bis [I-D.ietf-lisp-rfc6830bis], LISP-GPE [I-D.ietf-lisp-gpe], VXLAN [RFC7348], VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe], GRE [RFC2890], GTP [GTP-3GPP], ILA [I-D.herbert-intarea-ila], and Segment Routing (SRv6) [RFC8402].

Conceptually, LISP Map-Servers share some of the same basic configuration and maintenance properties as Domain Name System (DNS) [RFC1035] servers; likewise, Map-Resolvers are conceptually similar to DNS caching resolvers. With this in mind, this specification borrows familiar terminology (resolver and server) from the DNS specifications.

Note this document doesn't assume any particular database mapping infrastructure to illustrate certain aspects of Map-Server and Map-Resolver operation. The Mapping Service interface can (and likely will) be used by ITRs and ETRs to access other mapping database systems as the LISP infrastructure evolves.

LISP is not intended to address problems of connectivity and scaling on behalf of arbitrary communicating parties. Relevant situations are described in the scoping section of the introduction to [I-D.ietf-lisp-rfc6830bis].

This document obsoletes RFC 6830 and 6833.

1.1. Scope of Applicability

LISP was originally developed to address the Internet-wide route scaling problem [RFC4984]. While there are a number of approaches of interest for that problem, as LISP has been developed and refined, a large number of other LISP uses have been found and are being used. As such, the design and development of LISP has changed so as to focus on these use cases. The common property of these uses is a large set of cooperating entities seeking to communicate over the public Internet or other large underlay IP infrastructures, while keeping the addressing and topology of the cooperating entities separate from the underlay and Internet topology, routing, and addressing.

When communicating over the public Internet, deployers MUST consider the following guidelines:

1. LISP-SEC MUST be implemented [I-D.ietf-lisp-sec]. This means that the S-bit MUST be set in the Map-Reply (Section 5.4), Map-Register (Section 5.6) and Encapsulated Control messages (Section 5.8).
2. Implementations SHOULD use the 'HMAC-SHA256-128+HKDF-SHA256' as the Algorithm ID (Section 12.5) in Map-Register message (Section 5.6), and MUST NOT use 'None' or 'HMAC-SHA-1-96-None' as Algorithm ID (Section 12.5) in the Map-Register message (Section 5.6)

2. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Definition of Terms

Map-Server: A network infrastructure component that learns of EID-Prefix mapping entries from an ETR, via the registration mechanism described below, or some other authoritative source if one exists. A Map-Server publishes these EID-Prefixes in a mapping database.

Map-Request: A LISP Map-Request is a Control-Plane message to query the mapping system to resolve an EID. A LISP Map-Request can also be sent to an RLOC to test for reachability and to exchange security keys between an encapsulator and a decapsulator. This type of Map-Request is also known as an RLOC-Probe Request.

Map-Reply: A LISP Map-Reply is a Control-Plane message returned in response to a Map-Request sent to the mapping system when resolving an EID. A LISP Map-Reply can also be returned by a decapsulator in response to a Map-Request sent by an encapsulator to test for reachability. This type of Map-Reply is known as a RLOC-Probe Reply.

Encapsulated Map-Request: A LISP Map-Request carried within an Encapsulated Control Message (ECM), which has an additional LISP header prepended. Sent to UDP destination port 4342. The "outer" addresses are routable IP addresses, also known as RLOCs. Used by an ITR when sending to a Map-Resolver and by a Map-Server when forwarding a Map-Request to an ETR.

Map-Resolver: A network infrastructure component that accepts LISP Encapsulated (ECM) Map-Requests, typically from an ITR, and determines whether or not the destination IP address is part of the EID namespace; if it is not, a Negative Map-Reply is returned. Otherwise, the Map-Resolver finds the appropriate EID-to-RLOC mapping by consulting a mapping database system.

Negative Map-Reply: A LISP Map-Reply that contains an empty Locator-Set. Returned in response to a Map-Request if the destination EID is not registered in the mapping system, is policy denied or fails authentication.

Map-Register message: A LISP message sent by an ETR to a Map-Server to register its associated EID-Prefixes. In addition to the set of EID-Prefixes to register, the message includes one or more RLOCs to reach ETR(s). The Map-Server uses these RLOCs when forwarding Map-Requests (re-formatted as Encapsulated Map-Requests). An ETR MAY request that the Map-Server answer Map-Requests on its behalf by setting the "proxy Map-Reply" flag (P-bit) in the message.

Map-Notify message: A LISP message sent by a Map-Server to an ETR to confirm that a Map-Register has been received and processed. An ETR requests that a Map-Notify be returned by setting the "want-map-notify" flag (M-bit) in the Map-Register message. Unlike a Map-Reply, a Map-Notify uses UDP port 4342 for both source and destination. Map-Notify messages are also sent to ITRs by Map-Servers when there are RLOC-set changes.

For definitions of other terms, notably Ingress Tunnel Router (ITR), Egress Tunnel Router (ETR), and Re-encapsulating Tunnel Router (RTR), refer to the LISP Data-Plane specification [I-D.ietf-lisp-rfc6830bis].

4. Basic Overview

A Map-Server is a device that publishes EID-Prefixes in a LISP mapping database on behalf of a set of ETRs. When it receives a Map Request (typically originating from an ITR), it consults the mapping database to find an ETR that can answer with the set of RLOCs for an EID-Prefix. To publish its EID-Prefixes, an ETR periodically sends Map-Register messages to the Map-Server. A Map-Register message contains a list of EID-Prefixes plus a set of RLOCs that can be used to reach the ETRs.

When LISP-ALT [RFC6836] is used as the mapping database, a Map-Server connects to the ALT network and acts as a "last-hop" ALT-Router. Intermediate ALT-Routers forward Map-Requests to the Map-Server that advertises a particular EID-Prefix, and the Map-Server forwards them to the owning ETR, which responds with Map-Reply messages.

When LISP-DDT [RFC8111] is used as the mapping database, a Map-Server sends the final Map-Referral messages from the Delegated Database Tree.

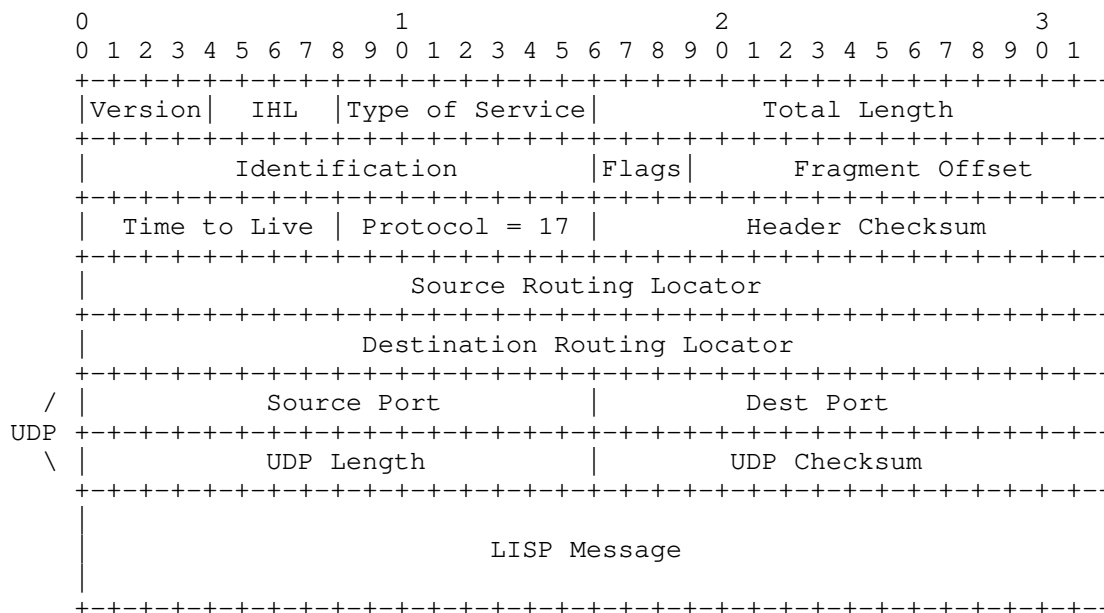
A Map-Resolver receives Encapsulated Map-Requests from its client ITRs and uses a mapping database system to find the appropriate ETR to answer those requests. On a LISP-ALT network, a Map-Resolver acts as a "first-hop" ALT-Router. It has Generic Routing Encapsulation (GRE) tunnels configured to other ALT-Routers and uses BGP to learn paths to ETRs for different prefixes in the LISP-ALT database. The Map-Resolver uses this path information to forward Map-Requests over the ALT to the correct ETRs. On a LISP-DDT network [RFC8111], a Map-Resolver maintains a referral-cache and acts as a "first-hop" DDT-node. The Map-Resolver uses the referral information to forward Map-Requests.

Note that while it is conceivable that a Map-Resolver could cache responses to improve performance, issues surrounding cache management would need to be resolved so that doing so will be reliable and practical. In this specification, Map-Resolvers will operate only in a non-caching mode, decapsulating and forwarding Encapsulated Map Requests received from ITRs. Any specification of caching functionality is out of scope for this document.

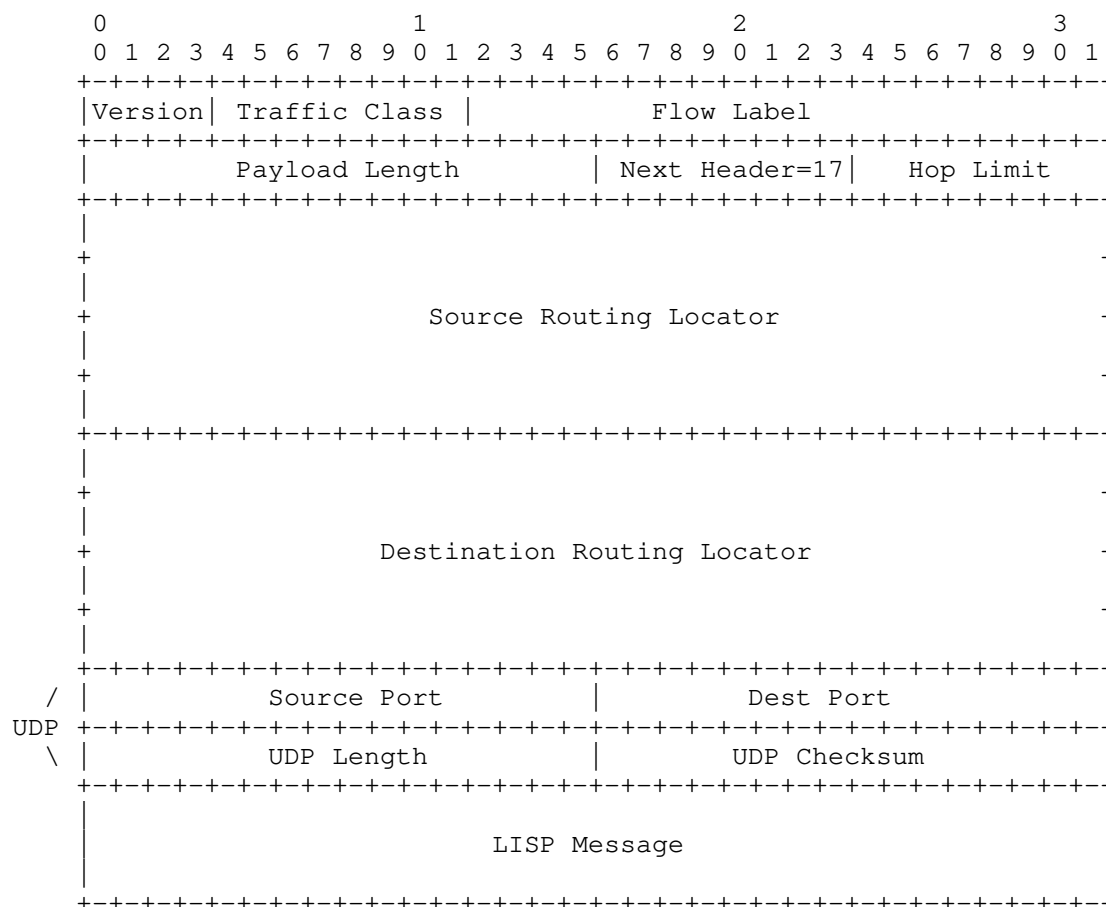
Note that a single device can implement the functions of both a Map-Server and a Map-Resolver, and in many cases the functions will be co-located in that way. Also, there can be ALT-only nodes and DDT-only nodes, when LISP-ALT and LISP-DDT are used, respectively, to connecting Map-Resolvers and Map-Servers together to make up the Mapping System.

5. LISP IPv4 and IPv6 Control-Plane Packet Formats

The following UDP packet formats are used by the LISP control plane.



IPv4 UDP LISP Control Message



IPv6 UDP LISP Control Message

When a UDP Map-Request, Map-Register, or Map-Notify (when used as a notification message) are sent, the UDP source port is chosen by the sender and the destination UDP port number is set to 4342. When a UDP Map-Reply, Map-Notify (when used as an acknowledgement to a Map-Register), or Map-Notify-Ack are sent, the source UDP port number is set to 4342 and the destination UDP port number is copied from the source port of either the Map-Request or the invoking data packet. Implementations MUST be prepared to accept packets when either the source port or destination UDP port is set to 4342 due to NATs changing port number values.

The 'UDP Length' field will reflect the length of the UDP header and the LISP Message payload. LISP is expected to be deployed by cooperating entities communicating over underlays. Deployers are

expected to set the MTU according to the specific deployment guidelines to prevent fragmentation of either the inner packet or the outer encapsulated packet. For deployments not aware of the underlay restrictions on path MTU, the message size MUST be limited to 576 bytes for IPv4 or 1280 bytes for IPv6 -considering the entire IP packet- as outlined in [RFC8085].

The UDP checksum is computed and set to non-zero for all messages sent to or from port 4342. It MUST be checked on receipt, and if the checksum fails, the control message MUST be dropped [RFC1071].

The format of control messages includes the UDP header so the checksum and length fields can be used to protect and delimit message boundaries.

5.1. LISP Control Packet Type Allocations

This section defines the LISP control message formats and summarizes for IANA the LISP Type codes assigned by this document. For completeness, the summary below includes the LISP Shared Extension Message assigned by [I-D.ietf-lisp-rfc8113bis]. Message type definitions are:

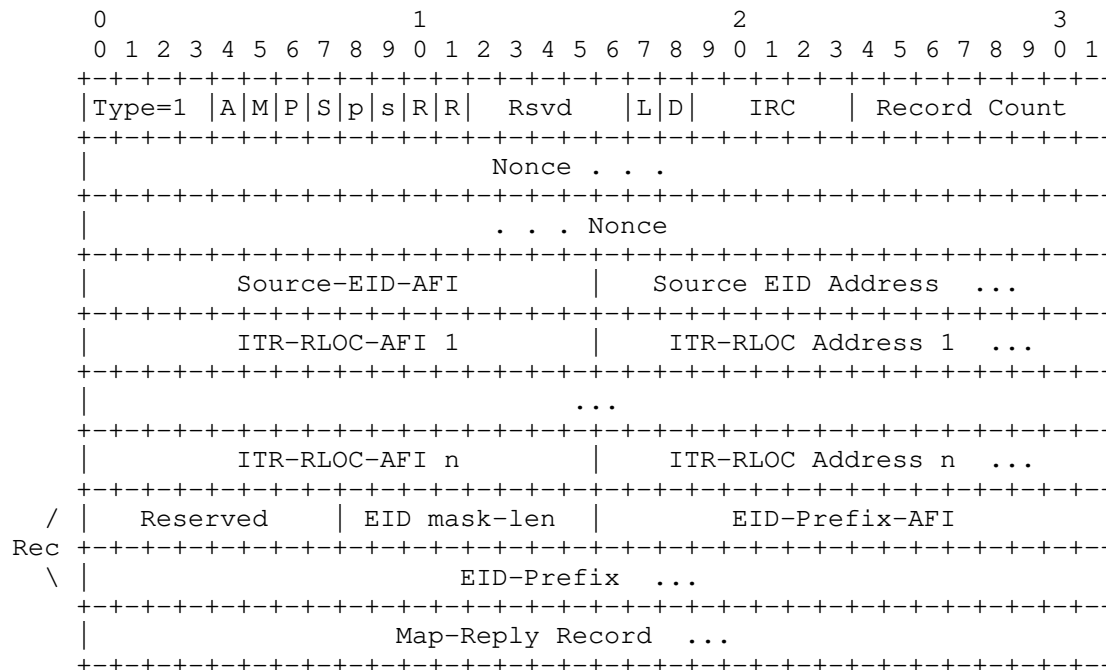
Reserved:	0	b'0000'
LISP Map-Request:	1	b'0001'
LISP Map-Reply:	2	b'0010'
LISP Map-Register:	3	b'0011'
LISP Map-Notify:	4	b'0100'
LISP Map-Notify-Ack:	5	b'0101'
LISP Map-Referral:	6	b'0110'
Unassigned	7	b'0111'
LISP Encapsulated Control Message:	8	b'1000'
Unassigned	9-14	b'1001' - b'1110'
LISP Shared Extension Message:	15	b'1111'

Protocol designers experimenting with new message formats are recommended to use the LISP Shared Extension Message Type described in [I-D.ietf-lisp-rfc8113bis].

All LISP Control-Plane messages use Address Family Identifiers (AFI) [AFI] or LISP Canonical Address Format (LCAF) [RFC8060] formats to encode either fixed or variable length addresses. This includes explicit fields in each control message or part of EID-records or RLOC-records in commonly formatted messages. LISP control-plane messages that include an unrecognized AFI MUST be dropped and the event MUST be logged.

The LISP control-plane describes how other data-planes can encode messages to support the Soliciting of Map-Requests as well as RLOC-probing procedures.

5.2. Map-Request Message Format



Packet field descriptions:

Type: 1 (Map-Request)

A: This is an authoritative bit, it is set to 1 when an ITR wants the destination site to return the Map-Reply rather than the mapping database system returning a Map-Reply, and set to 0 otherwise.

M: This is the map-data-present bit. When set, it indicates that a Map-Reply Record segment is included in the Map-Request.

P: This is the probe-bit, which indicates that a Map-Request MUST be treated as a Locator reachability probe. The receiver MUST respond with a Map-Reply with the probe-bit set, indicating that the Map-Reply is a Locator reachability probe reply, with the nonce copied from the Map-Request. See RLOC-Probing Section 7.1 for more details. This RLOC-probe Map-Request MUST NOT be sent to the mapping system. If a Map-Resolver or Map-Server receives a Map-Request with the probe-bit set, it MUST drop the message.

S: This is the Solicit-Map-Request (SMR) bit. See Solicit-Map-Request (SMRs) Section 6.1 for details.

- p: This is the PITR bit. This bit is set to 1 when a PITR sends a Map-Request. The use of this bit is deployment-specific.
- s: This is the SMR-invoked bit. This bit is set to 1 when an xTR is sending a Map-Request in response to a received SMR-based Map-Request.
- R: This reserved and unassigned bit MUST be set to 0 on transmit and MUST be ignored on receipt.
- Rsvd: This field MUST be set to 0 on transmit and MUST be ignored on receipt.
- L: This is the local-xtr bit. It is used by an xTR in a LISP site to tell other xTRs in the same site that it is part of the RLOC-set for the LISP site. The L-bit is set to 1 when the RLOC is the sender's IP address.
- D: This is the dont-map-reply bit. It is used in the SMR procedure described in Section 6.1. When an xTR sends an SMR message, it doesn't need a Map-Reply returned. When this bit is set, the receiver of the Map-Request does not return a Map-Reply.
- IRC: This 5-bit field is the ITR-RLOC Count, which encodes the additional number of ('ITR-RLOC-AFI', 'ITR-RLOC Address') fields present in this message. At least one (ITR-RLOC-AFI, ITR-RLOC-Address) pair MUST be encoded. Multiple 'ITR-RLOC Address' fields are used, so a Map-Replier can select which destination address to use for a Map-Reply. The IRC value ranges from 0 to 31. For a value of 0, there is 1 ITR-RLOC address encoded; for a value of 1, there are 2 ITR-RLOC addresses encoded, and so on up to 31, which encodes a total of 32 ITR-RLOC addresses.
- Record Count: This is the number of records in this Map-Request message. A record is comprised of the portion of the packet that is labeled 'Rec' above and occurs the number of times equal to Record Count. For this version of the protocol, a receiver MUST accept and process Map-Requests that contain one or more records, but a sender MUST only send Map-Requests containing one record.
- Nonce: This is an 8-octet random value created by the sender of the Map-Request. This nonce will be returned in the Map-Reply. The nonce is used as an index to identify the corresponding Map-Request when a Map-Reply message is received. The nonce MUST be generated by a properly seeded pseudo-random source, see as an example [RFC4086].

Source-EID-AFI: This is the address family of the 'Source EID Address' field.

Source EID Address: This is the EID of the source host that originated the packet that caused the Map-Request. When Map-Requests are used for refreshing a Map-Cache entry or for RLOC-Probing, an AFI value 0 is used and this field is of zero length.

ITR-RLOC-AFI: This is the address family of the 'ITR-RLOC Address' field that follows this field.

ITR-RLOC Address: This is used to give the ETR the option of selecting the destination address from any address family for the Map-Reply message. This address MUST be a routable RLOC address of the sender of the Map-Request message.

EID mask-len: This is the mask length for the EID-Prefix.

EID-Prefix-AFI: This is the address family of the EID-Prefix according to [AFI] and [RFC8060].

EID-Prefix: This prefix address length is 4 octets for an IPv4 address family and 16 octets for an IPv6 address family when the EID-Prefix-AFI is 1 or 2, respectively. For other AFIs [AFI], the address length varies and for the LCAF AFI the format is defined in [RFC8060]. When a Map-Request is sent by an ITR because a data packet is received for a destination where there is no mapping entry, the EID-Prefix is set to the destination IP address of the data packet, and the 'EID mask-len' is set to 32 or 128 for IPv4 or IPv6, respectively. When an xTR wants to query a site about the status of a mapping it already has cached, the EID-Prefix used in the Map-Request has the same mask-length as the EID-Prefix returned from the site when it sent a Map-Reply message.

Map-Reply Record: When the M-bit is set, this field is the size of a single "Record" in the Map-Reply format. This Map-Reply record contains the EID-to-RLOC mapping entry associated with the Source EID. This allows the ETR that will receive this Map-Request to cache the data if it chooses to do so. It is important to note that this mapping has not been validated by the Mapping System.

5.3. EID-to-RLOC UDP Map-Request Message

A Map-Request is sent from an ITR when it needs a mapping for an EID, wants to test an RLOC for reachability, or wants to refresh a mapping before TTL expiration. For the initial case, the destination IP address used for the Map-Request is the data packet's destination address (i.e., the destination EID) that had a mapping cache lookup

failure. For the latter two cases, the destination IP address used for the Map-Request is one of the RLOC addresses from the Locator-Set of the Map-Cache entry. The source address is either an IPv4 or IPv6 RLOC address, depending on whether the Map-Request is using an IPv4 or IPv6 header, respectively. In all cases, the UDP source port number for the Map-Request message is a 16-bit value selected by the ITR/PITR, and the UDP destination port number is set to the well-known destination port number 4342. A successful Map-Reply, which is one that has a nonce that matches an outstanding Map-Request nonce, will update the cached set of RLOCs associated with the EID-Prefix range.

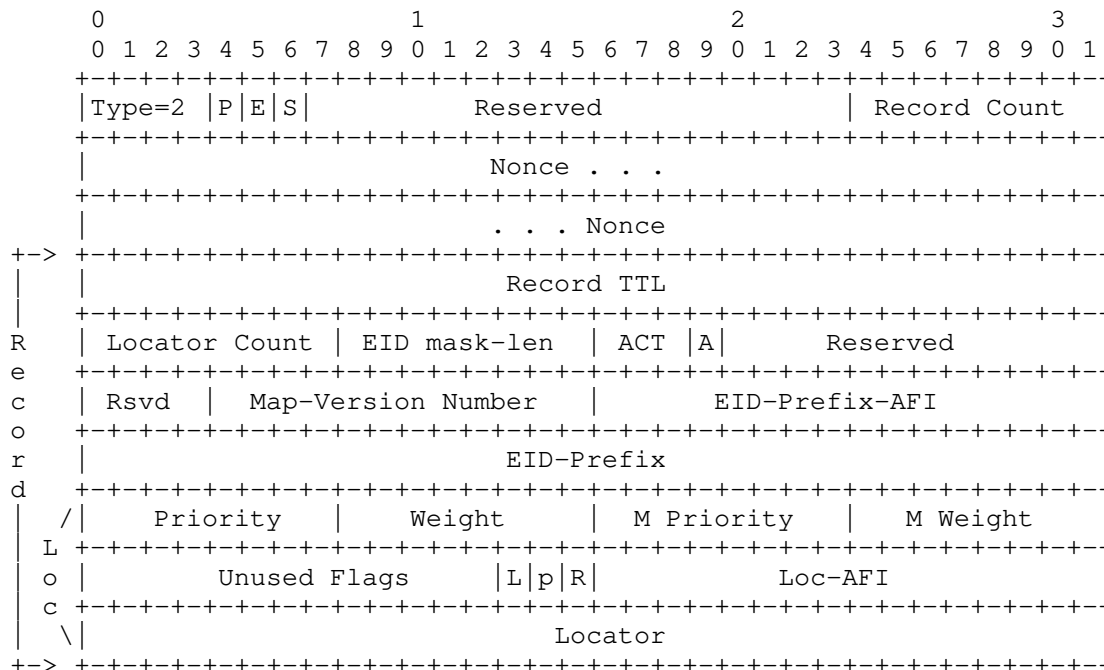
One or more Map-Request ('ITR-RLOC-AFI', 'ITR-RLOC-Address') fields MUST be filled in by the ITR. The number of fields (minus 1) encoded MUST be placed in the 'IRC' field. The ITR MAY include all locally configured Locators in this list or just provide one locator address from each address family it supports. If the ITR erroneously provides no ITR-RLOC addresses, the Map-Replier MUST drop the Map-Request.

Map-Requests can also be LISP encapsulated using UDP destination port 4342 with a LISP Type value set to "Encapsulated Control Message", when sent from an ITR to a Map-Resolver. Likewise, Map-Requests are LISP encapsulated the same way from a Map-Server to an ETR. Details on Encapsulated Map-Requests and Map-Resolvers can be found in Section 5.8.

Map-Requests MUST be rate-limited to 1 per second per EID-prefix. After 10 retransmits without receiving the corresponding Map-Reply the sender MUST wait 30 seconds.

An ITR that is configured with mapping database information (i.e., it is also an ETR) MAY optionally include those mappings in a Map-Request. When an ETR configured to accept and verify such "piggybacked" mapping data receives such a Map-Request and it does not have this mapping in the Map-Cache, it MUST originate a "verifying Map-Request" through the mapping database to validate the "piggybacked" mapping data.

5.4. Map-Reply Message Format



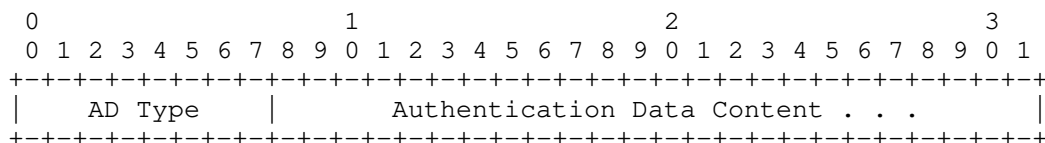
Packet field descriptions:

Type: 2 (Map-Reply)

P: This is the probe-bit, which indicates that the Map-Reply is in response to a Locator reachability probe Map-Request. The 'Nonce' field must contain a copy of the nonce value from the original Map-Request. See RLOC-probing Section 7.1 for more details. When the probe-bit is set to 1 in a Map-Reply message, the A-bit in each EID-record included in the message MUST be set to 1, otherwise MUST be silently discarded.

E: This bit indicates that the ETR that sends this Map-Reply message is advertising that the site is enabled for the Echo-Nonce Locator reachability algorithm. See Echo-Nonce [I-D.ietf-lisp-rfc6830bis] for more details.

S: This is the Security bit. When set to 1, the following authentication information will be appended to the end of the Map-Reply. The details can be found in [I-D.ietf-lisp-sec].



Reserved: This unassigned field MUST be set to 0 on transmit and MUST be ignored on receipt.

Record Count: This is the number of records in this reply message. A record is comprised of that portion of the packet labeled 'Record' above and occurs the number of times equal to Record Count. Note that the reply count can be larger than the requested count, for instance when more-specifics are present.

Nonce: This 64-bit value from the Map-Request is echoed in this 'Nonce' field of the Map-Reply.

Record TTL: This is the time in minutes the recipient of the Map-Reply can store the mapping. If the TTL is 0, the entry MUST be removed from the cache immediately. If the value is 0xffffffff, the recipient can decide locally how long to store the mapping.

Locator Count: This is the number of Locator entries in the given Record. A Locator entry comprises what is labeled above as 'Loc'. The Locator count can be 0, indicating that there are no Locators for the EID-Prefix.

EID mask-len: This is the mask length for the EID-Prefix.

ACT: This 3-bit field describes Negative Map-Reply actions. In any other message type, these bits are set to 0 and ignored on receipt. These bits are used only when the 'Locator Count' field is set to 0. The action bits are encoded only in Map-Reply messages. They are used to tell an ITR or PITR why a empty locator-set was returned from the mapping system and how it stores the map-cache entry. See Section 12.3 for additional information.

(0) No-Action: The Map-Cache is kept alive, and no packet encapsulation occurs.

(1) Natively-Forward: The packet is not encapsulated or dropped but natively forwarded.

- (2) Send-Map-Request: The Map-Cache entry is created and flagged that any packet matching this entry invokes sending a Map-Request.
 - (3) Drop/No-Reason: A packet that matches this Map-Cache entry is dropped. An ICMP Destination Unreachable message SHOULD be sent.
 - (4) Drop/Policy-Denied: A packet that matches this Map-Cache entry is dropped. The reason for the Drop action is that a Map-Request for the target-EID is being policy denied by either an xTR or the mapping system.
 - (5) Drop/Authentication-Failure: A packet that matches this Map-Cache entry is dropped. The reason for the Drop action is that a Map-Request for the target-EID fails an authentication verification-check by either an xTR or the mapping system.
- A: The Authoritative bit MAY only be set to 1 by an ETR. A Map-Server generating Map-Reply messages as a proxy MUST NOT set the A-bit to 1. This bit indicates to the requesting ITRs if the Map-Reply was originated by a LISP node managed at the site that owns the EID-Prefix.

Map-Version Number: When this 12-bit value in an EID-record of a Map-Reply message is non-zero, follow the procedures in [I-D.ietf-lisp-6834bis] for details.

EID-Prefix-AFI: Address family of the EID-Prefix according to [AFI] and [RFC8060].

EID-Prefix: This prefix is 4 octets for an IPv4 address family and 16 octets for an IPv6 address family.

Priority: Each RLOC is assigned a unicast Priority. Lower values are more preferable. When multiple RLOCs have the same Priority, they may be used in a load-split fashion. A value of 255 means the RLOC MUST NOT be used for unicast forwarding.

Weight: When priorities are the same for multiple RLOCs, the Weight indicates how to balance unicast traffic between them. Weight is encoded as a relative weight of total unicast packets that match the mapping entry. For example, if there are 4 Locators in a Locator-Set, where the Weights assigned are 30, 20, 20, and 10, the first Locator will get 37.5% of the traffic, the 2nd and 3rd Locators will each get 25% of the traffic, and the 4th Locator will get 12.5% of the traffic. If all Weights for a Locator-Set are equal, the receiver of the Map-Reply will decide how to load-

split the traffic. See RLOC-hashing [I-D.ietf-lisp-rfc6830bis] for a suggested hash algorithm to distribute the load across Locators with the same Priority and equal Weight values.

M Priority: Each RLOC is assigned a multicast Priority used by an ETR in a receiver multicast site to select an ITR in a source multicast site for building multicast distribution trees. A value of 255 means the RLOC MUST NOT be used for joining a multicast distribution tree. For more details, see [RFC6831].

M Weight: When priorities are the same for multiple RLOCs, the Weight indicates how to balance building multicast distribution trees across multiple ITRs. The Weight is encoded as a relative weight (similar to the unicast Weights) of the total number of trees built to the source site identified by the EID-Prefix. If all Weights for a Locator-Set are equal, the receiver of the Map-Reply will decide how to distribute multicast state across ITRs. For more details, see [RFC6831].

Unused Flags: These are set to 0 when sending and ignored on receipt.

L: When this bit is set, the Locator is flagged as a local Locator to the ETR that is sending the Map-Reply. When a Map-Server is doing proxy Map-Replying for a LISP site, the L-bit is set to 0 for all Locators in this Locator-Set.

p: When this bit is set, an ETR informs the RLOC-Probing ITR that the locator address for which this bit is set is the one being RLOC-probed and may be different from the source address of the Map-Reply. An ITR that RLOC-probes a particular Locator MUST use this Locator for retrieving the data structure used to store the fact that the Locator is reachable. The p-bit is set for a single Locator in the same Locator-Set. If an implementation sets more than one p-bit erroneously, the receiver of the Map-Reply MUST select the first set p-bit Locator. The p-bit MUST NOT be set for Locator-Set records sent in Map-Request and Map-Register messages.

R: This is set when the sender of a Map-Reply has a route to the Locator in the Locator data record. This receiver may find this useful to know if the Locator is up but not necessarily reachable from the receiver's point of view.

Locator: This is an IPv4 or IPv6 address (as encoded by the 'Loc-AFI' field) assigned to an ETR and used by an ITR as a destination RLOC address in the outer header of a LISP encapsulated packet. Note that the destination RLOC address of a LISP encapsulated packet MAY be an anycast address. A source RLOC of a LISP

encapsulated packet can be an anycast address as well. The source or destination RLOC MUST NOT be the broadcast address (255.255.255.255 or any subnet broadcast address known to the router) and MUST NOT be a link-local multicast address. The source RLOC MUST NOT be a multicast address. The destination RLOC SHOULD be a multicast address if it is being mapped from a multicast destination EID.

Map-Reply MUST be rate-limited, it is RECOMMENDED that a Map-Reply for the same destination RLOC be sent no more than one packets per 3 seconds.

The Record format, as defined here, is used both in the Map-Reply and Map-Register messages, this includes all the field definitions.

5.5. EID-to-RLOC UDP Map-Reply Message

A Map-Reply returns an EID-Prefix with a mask-length that is less than or equal to the EID being requested. The EID being requested is either from the destination field of an IP header of a Data-Probe or the EID of a record of a Map-Request. The RLOCs in the Map-Reply are routable IP addresses of all ETRs for the LISP site. Each RLOC conveys status reachability but does not convey path reachability from a requester's perspective. Separate testing of path reachability is required. See RLOC-reachability Section 7.1 for details.

Note that a Map-Reply MAY contain different EID-Prefix granularity (prefix + mask-length) than the Map-Request that triggers it. This might occur if a Map-Request were for a prefix that had been returned by an earlier Map-Reply. In such a case, the requester updates its cache with the new prefix information and granularity. For example, a requester with two cached EID-Prefixes that are covered by a Map-Reply containing one less-specific prefix replaces the entry with the less-specific EID-Prefix. Note that the reverse, replacement of one less-specific prefix with multiple more-specific prefixes, can also occur, not by removing the less-specific prefix but rather by adding the more-specific prefixes that, during a lookup, will override the less-specific prefix.

When an EID moves out of a LISP site [I-D.ietf-lisp-eid-mobility], the database mapping system may have overlapping EID-prefixes. Or when a LISP site is configured with multiple sets of ETRs that support different EID-prefix mask-lengths, the database mapping system may have overlapping EID-prefixes. When overlapping EID-prefixes exist, a Map-Request with an EID that best matches any EID-Prefix MUST be returned in a single Map-Reply message. For instance, if an ETR had database mapping entries for EID-Prefixes:

```
2001:db8::/32
2001:db8:1::/48
2001:db8:1:1::/64
2001:db8:1:2::/64
```

A Map-Request for EID 2001:db8:1:1::1 would cause a Map-Reply with a record count of 1 to be returned with a mapping record EID-Prefix of 2001:db8:1:1::/64.

A Map-Request for EID 2001:db8:1:5::5 would cause a Map-Reply with a record count of 3 to be returned with mapping records for EID-Prefixes 2001:db8:1::/48, 2001:db8:1:1::/64, 2001:db8:1:2::/64, filling out the /48 with more-specifics that exist in the mapping system.

Note that not all overlapping EID-Prefixes need to be returned but only the more-specific entries (note that in the second example above 2001:db8::/32 was not returned for requesting EID 2001:db8:1:5::5) for the matching EID-Prefix of the requesting EID. When more than one EID-Prefix is returned, all SHOULD use the same Time to Live value so they can all time out at the same time. When a more-specific EID-Prefix is received later, its Time to Live value in the Map-Reply record can be stored even when other less-specific entries exist. When a less-specific EID-Prefix is received later, its Map-Cache expiration time SHOULD be set to the minimum expiration time of any more-specific EID-Prefix in the Map-Cache. This is done so the integrity of the EID-Prefix set is wholly maintained and so no more-specific entries are removed from the Map-Cache while keeping less-specific entries.

For scalability, it is expected that aggregation of EID addresses into EID-Prefixes will allow one Map-Reply to satisfy a mapping for the EID addresses in the prefix range, thereby reducing the number of Map-Request messages.

Map-Reply records can have an empty Locator-Set. A Negative Map-Reply is a Map-Reply with an empty Locator-Set. Negative Map-Replies convey special actions by the sender to the ITR or PITR that have solicited the Map-Reply. There are two primary applications for Negative Map-Replies. The first is for a Map-Resolver to instruct an ITR or PITR when a destination is for a LISP site versus a non-LISP site, and the other is to source quench Map-Requests that are sent for non-allocated EIDs.

For each Map-Reply record, the list of Locators in a Locator-Set MUST be sorted in order of ascending IP address where an IPv4 locator address is considered numerically 'less than' an IPv6 locator address.

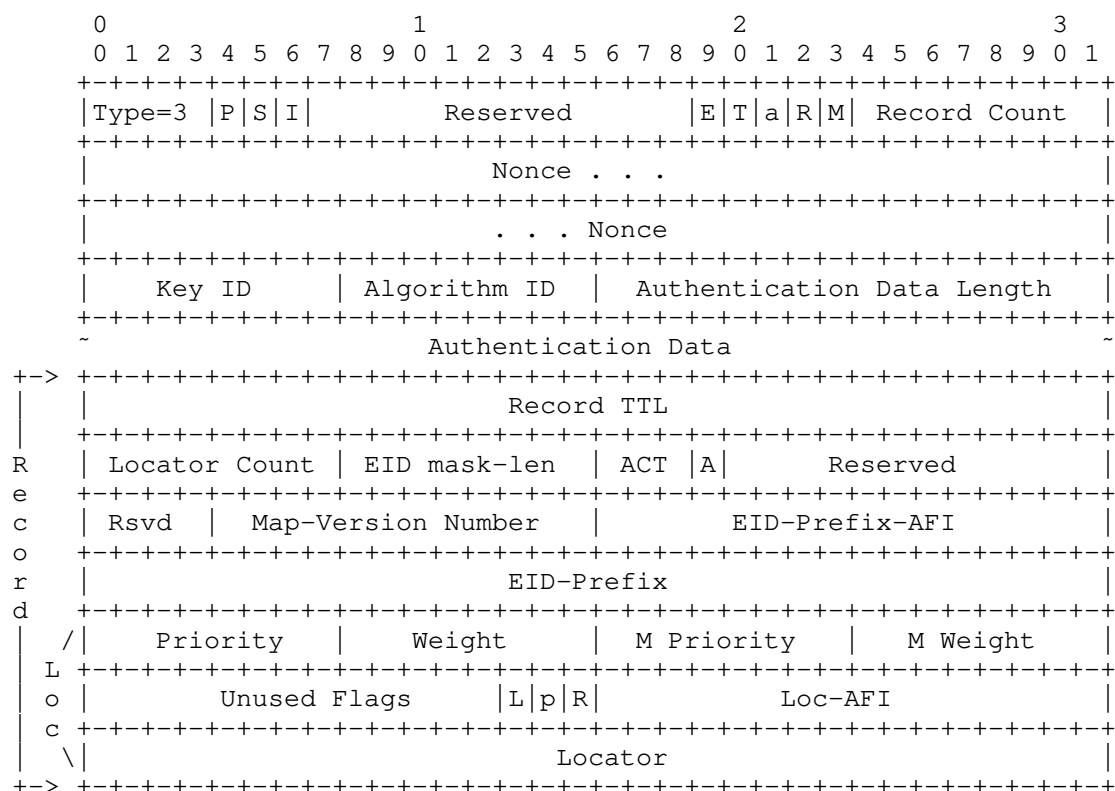
When sending a Map-Reply message, the destination address is copied from one of the 'ITR-RLLOC' fields from the Map-Request. The ETR can choose a locator address from one of the address families it supports. For Data-Probes, the destination address of the Map-Reply is copied from the source address of the Data-Probe message that is invoking the reply. The source address of the Map-Reply is one of the local IP addresses chosen, to allow Unicast Reverse Path Forwarding (uRPF) checks to succeed in the upstream service provider. The destination port of a Map-Reply message is copied from the source port of the Map-Request or Data-Probe, and the source port of the Map-Reply message is set to the well-known UDP port 4342.

5.6. Map-Register Message Format

This section specifies the encoding format for the Map-Register message. The message is sent in UDP with a destination UDP port of 4342 and a randomly selected UDP source port number.

The fields below are used in multiple control messages. They are defined for Map-Register, Map-Notify and Map-Notify-Ack message types.

The Map-Register message format is:



Packet field descriptions:

Type: 3 (Map-Register)

P: This is the proxy Map-Reply bit. When set to 1, the ETR sending the Map-Register message is requesting the Map-Server to proxy a Map-Reply. The Map-Server will send non-authoritative Map-Replies on behalf of the ETR.

- S: This is the security-capable bit. When set, the procedures from [I-D.ietf-lisp-sec] are supported.
- I: This is the ID-present bit. This bit is set to 1 to indicate that a 128 bit xTR-ID and a 64 bit Site-ID fields are present at the end of the Map-Register message. If an xTR is configured with an xTR-ID and Site-ID, it MUST set the I bit to 1 and include its xTR-ID and Site-ID in the Map-Register messages it generates. The combination of Site-ID plus xTR-ID uniquely identifies an xTR in a LISP domain and serves to track its last seen nonce.
- Reserved: This unassigned field MUST be set to 0 on transmit and MUST be ignored on receipt.
- E: This is the Map-Register EID-notify bit. This is used by a First-Hop-Router (FHR) which discovers a dynamic-EID. This EID-notify based Map-Register is sent by the FHR to a same site xTR that propagates the Map-Register to the mapping system. The site xTR keeps state to later Map-Notify the FHR after the EID has moves away. See [I-D.ietf-lisp-eid-mobility] for a detailed use-case.
- T: This is the use-TTL for timeout bit. When set to 1, the xTR wants the Map-Server to time out registrations based on the value in the "Record TTL" field of this message. Otherwise, the default timeout described in Section 8.2 is used.
- a: This is the merge-request bit. When set to 1, the xTR requests to merge RLOC-records from different xTRs registering the same EID-record. See signal-free multicast [RFC8378] for one use case example.
- R: This reserved and unassigned bit MUST be set to 0 on transmit and MUST be ignored on receipt.
- M: This is the want-map-notify bit. When set to 1, an ETR is requesting a Map-Notify message to be returned in response to sending a Map-Register message. The Map-Notify message sent by a Map-Server is used to acknowledge receipt of a Map-Register message.
- Record Count: This is the number of records in this Map-Register message. A record is comprised of that portion of the packet labeled 'Record' above and occurs the number of times equal to Record Count.
- Nonce: This 8-octet 'Nonce' field is incremented each time a Map-Register message is sent. When a Map-Register acknowledgement is requested, the nonce is returned by Map-Servers in Map-Notify

messages. Since the entire Map-Register message is authenticated, the 'Nonce' field serves to protect against Map-Register replay attacks. An ETR that registers to the mapping system SHOULD store the last nonce sent in persistent storage so when it restarts it can continue using an incrementing nonce. If the ETR cannot support saving the nonce, then when it restarts it MUST use a new authentication key to register to the mapping system. A Map-Server MUST track and save in persistent storage the last nonce received for each ETR xTR-ID and key pair. If a Map-Register is received with a nonce value that is not greater than the saved nonce, it MUST drop the Map-Register message and SHOULD log the fact a replay attack could have occurred.

Key ID: A key-id value that identifies a pre-shared secret between an ETR and a Map-Server. Per-message keys are derived from the pre-shared secret to authenticate the origin and protect the integrity of the Map-Register. The Key ID allows to rotate between multiple pre-shared secrets in a non disruptive way. The pre-shared secret MUST be unique per each LISP "Site-ID"

Algorithm ID: This field identifies the Key Derivation Function (KDF) and Message Authentication Code (MAC) algorithms used to derive the key and to compute the Authentication Data of a Map-Register. This 8-bit field identifies the KDF and MAC algorithm pair. See Section 12.5 for codepoint assignments.

Authentication Data Length: This is the length in octets of the 'Authentication Data' field that follows this field. The length of the 'Authentication Data' field is dependent on the MAC algorithm used. The length field allows a device that doesn't know the MAC algorithm to correctly parse the packet.

Authentication Data: This is the output of the MAC algorithm placed in this field after the MAC computation. The MAC output is computed as follows:

- 1: The KDF algorithm is identified by the field 'Algorithm ID' according to the table in Section 12.5. Implementations of this specification MUST implement HMAC-SHA-256-128 [RFC4868] and SHOULD implement HMAC-SHA-256-128+HKDF-SHA256 [RFC5869] .
- 2: The MAC algorithm is identified by the field 'Algorithm ID' according to the table in Section 12.5.

- 3: The pre-shared secret used to derive the per-message key is represented by PSK[Key ID], that is the pre-shared secret identified by the 'Key ID'.
- 4: The derived per-message key is computed as: $\text{per-msg-key} = \text{KDF}(\text{nonce} + \text{PSK}[\text{Key ID}], s)$. Where the nonce is the value in the Nonce field of the Map-Register, '+' denotes concatenation and 's' (the salt) is a string that corresponds to the message type being authenticated. For Map-Register messages, it is equal to "Map-Register Authentication". Similarly, for Map-Notify and Map-Notify-Ack messages, it is "Map-Notify Authentication" and "Map-Notify-Ack Authentication", respectively. For those Algorithm IDs defined in Section 12.5 that specify a 'none' KDF, the per-message key is computed as: $\text{per-msg-key} = \text{PSK}[\text{Key ID}]$. This means that the same key is used across multiple protocol messages.
- 5: The MAC output is computed using the MAC algorithm and the per-msg-key over the entire Map-Register payload (from and including the LISP message type field through the end of the last RLOC record) with the authenticated data field preset to 0.

The definition of the rest of the Map-Register can be found in EID-record description in Section 5.4. When the I-bit is set, the following fields are added to the end of the Map-Register message:

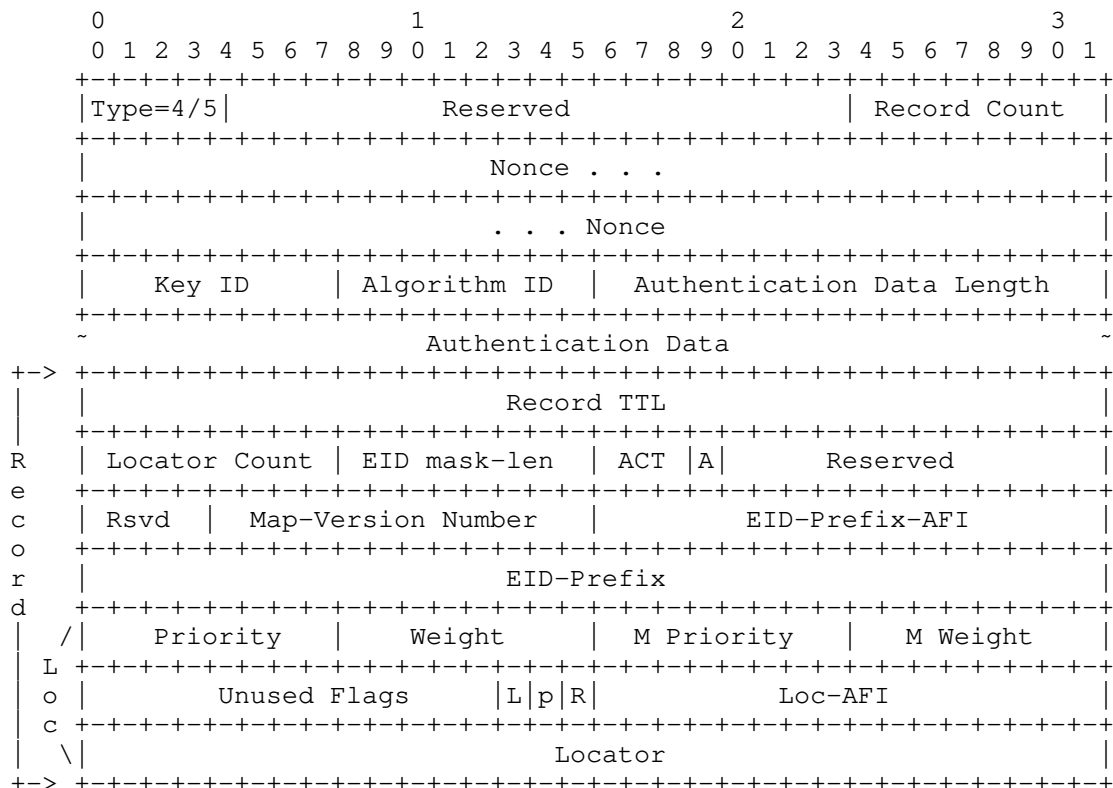
xTR-ID: xTR-ID is a 128 bit field at the end of the Map-Register message, starting after the final Record in the message. The xTR-ID is used to uniquely identify a xTR. The same xTR-ID value MUST NOT be used in two different xTRs in the scope of the Site-ID.

Site-ID: Site-ID is a 64 bit field at the end of the Map-Register message, following the xTR-ID. Site-ID is used to uniquely identify to which site the xTR that sent the message belongs. This document does not specify a strict meaning for the Site-ID field. Informally it provides an indication that a group of xTRs have some relation, either administratively, topologically or otherwise.

5.7. Map-Notify/Map-Notify-Ack Message Format

This section specifies the encoding format for the Map-Notify and Map-Notify-Ack messages. The messages are sent inside a UDP packet with source and destination UDP ports equal to 4342.

The Map-Notify and Map-Notify-Ack message formats are:



Packet field descriptions:

Type: 4/5 (Map-Notify/Map-Notify-Ack)

The Map-Notify message has the same contents as a Map-Register message. See the Map-Register section for field descriptions and the Map-Reply section for EID-record and RLOC-record descriptions.

The fields of the Map-Notify are copied from the corresponding Map-Register to acknowledge its correct processing. In the Map-Notify, the 'Authentication Data' field is recomputed using the corresponding per-message key and according to the procedure defined in the

previous section. The Map-Notify message can also be used, outside the scope of this specification, in an unsolicited manner, such as is specified in [I-D.ietf-lisp-pubsub].

After sending a Map-Register, if a Map-Notify is not received after 1 second the transmitter MUST re-transmit the original Map-Register with an exponential backoff (base of 2, that is, the next backoff timeout interval is doubled), the maximum backoff is 1 minute. Map-Notify messages are only transmitted upon the reception of a Map-Register with the M-bit set, Map-Notify messages are not retransmitted. The only exception to this is for unsolicited Map-Notify messages, see below.

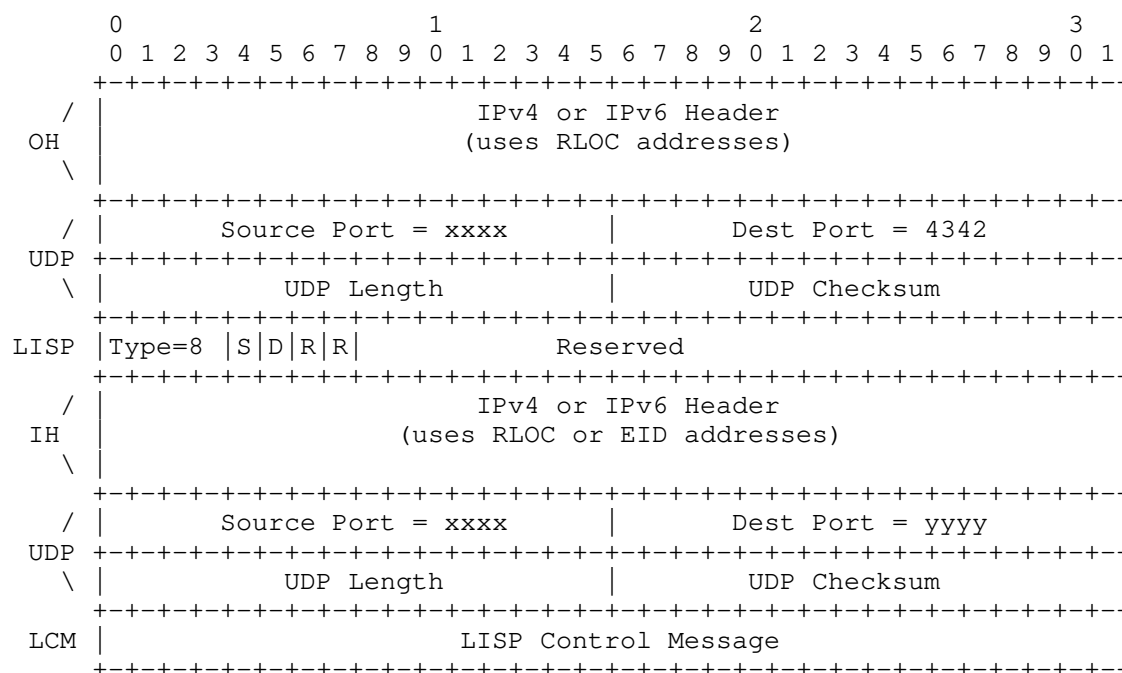
A Map-Server sends an unsolicited Map-Notify message (one that is not used as an acknowledgment to a Map-Register message) only in conformance with the Congestion Control And Reliability Guideline sections of [RFC8085]. A Map-Notify is retransmitted until a Map-Notify-Ack is received by the Map-Server with the same nonce used in the Map-Notify message. An implementation SHOULD retransmit up to 3 times at 3 second retransmission intervals, after which time the retransmission interval is exponentially backed-off (base of 2, that is, the next backoff timeout interval is doubled) for another 3 retransmission attempts. Map-Notify-Ack messages are only transmitted upon the reception of an unsolicited Map-Notify, Map-Notify-Ack messages are not retransmitted.

The Map-Notify-Ack message has the same contents as a Map-Notify message. It is used to acknowledge the receipt of an unsolicited Map-Notify and, once the authentication data is validated, allows for the sender to stop retransmitting a Map-Notify with the same nonce and the authentication data validates. The fields of the Map-Notify-Ack are copied from the corresponding Map-Notify message to acknowledge its correct processing. The 'Authentication Data' field is recomputed using the corresponding per-message key and according to the procedure defined in the previous section.

Upon reception of Map-Register, Map-Notify or Map-Notify-Ack, the receiver verifies the authentication data. If the authentication data fails to validate, the message is dropped without further processing.

5.8. Encapsulated Control Message Format

An Encapsulated Control Message (ECM) is used to encapsulate control packets sent between xTRs and the mapping database system or internal to the mapping database system.



Packet header descriptions:

OH: The outer IPv4 or IPv6 header, which uses RLOC addresses in the source and destination header address fields.

UDP: The outer UDP header with destination port 4342. The source port is randomly allocated. The checksum field **MUST** be non-zero.

LISP: Type 8 is defined to be a "LISP Encapsulated Control Message", and what follows is either an IPv4 or IPv6 header as encoded by the first 4 bits after the 'Reserved' field, or the Authentication Data field [I-D.ietf-lisp-sec] if the S-bit (see below) is set.

Type: 8 (Encapsulated Control Message (ECM))

S: This is the Security bit. When set to 1, the field following the 'Reserved' field will have the following Authentication Data format and follow the procedures from [I-D.ietf-lisp-sec].

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|      AD Type      |      Authentication Data Content . . .      |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

D: This is the DDT-bit. When set to 1, the sender is requesting a Map-Referral message to be returned. The details of this procedure are described in [RFC8111].

R: This reserved and unassigned bit MUST be set to 0 on transmit and MUST be ignored on receipt.

IH: The inner IPv4 or IPv6 header, which can use either RLOC or EID addresses in the header address fields. When a Map-Request is encapsulated in this packet format, the destination address in this header is an EID.

UDP: The inner UDP header, where the port assignments depend on the control packet being encapsulated. When the control packet is a Map-Request or Map-Register, the source port is selected by the ITR/PITR and the destination port is 4342. When the control packet is a Map-Reply, the source port is 4342 and the destination port is assigned from the source port of the invoking Map-Request. Port number 4341 MUST NOT be assigned to either port. The checksum field MUST be non-zero.

LCM: The format is one of the control message formats described in Section 5. Map-Request messages are allowed to be Control-Plane (ECM) encapsulated. When Map-Requests are sent for RLOC-Probing purposes (i.e. the probe-bit is set), they MUST NOT be sent inside Encapsulated Control Messages. PIM Join/Prune messages [RFC6831] are also allowed to be Control-Plane (ECM) encapsulated.

6. Changing the Contents of EID-to-RLOC Mappings

In the LISP architecture ITRs/PITRs use a local Map-Cache to store EID-to-RLOC mappings for forwarding. When an ETR updates a mapping a mechanism is required to inform ITRs/PITRs that are using such mappings.

The LISP Data-Plane defines several mechanism to update mappings [I-D.ietf-lisp-rfc6830bis]. This document specifies the Solicit-Map-Request (SMR), a Control-Plane push-based mechanism. An additional Control-Plane mechanism based on the Publish/subscribe paradigm is specified in [I-D.ietf-lisp-pubsub].

6.1. Solicit-Map-Request (SMR)

Soliciting a Map-Request is a selective way for ETRs, at the site where mappings change, to control the rate they receive requests for Map-Reply messages. SMRs are also used to tell remote ITRs to update the mappings they have cached.

Since ETRs are not required to keep track of remote ITRs that have cached their mappings, they do not know which ITRs need to have their mappings updated. As a result, an ETR will solicit Map-Requests to those sites to which it has been sending LISP encapsulated data packets for the last minute. As a result, when an ETR is also acting as ITR, it will send an SMR to an ITR to which it has recently sent encapsulated data.

An SMR message is simply a bit set in a Map-Request message. An ITR or PITR will send a Map-Request (SMR-invoked Map-Request) when they receive an SMR message. While the SMR message is sent through the data-plane, the SMR-invoked Map-Request MUST be sent through the Mapping System (not directly).

Both the SMR sender and the SMR responder MUST rate-limit these messages. It is RECOMMENDED that the SMR sender rate-limits Map-Request for the same destination RLOC to no more than one packet per 3 seconds. It is RECOMMENDED that the SMR responder rate-limits Map-Request for the same EID-Prefix to no more than once per 3 seconds.

When an ITR receives an SMR message for which it does not have a cached mapping for the EID in the SMR message, it SHOULD NOT send an SMR-invoked Map-Request. This scenario can occur when an ETR sends SMR messages to all Locators in the Locator-Set it has stored in its Map-Cache but the remote ITRs that receive the SMR may not be sending packets to the site. There is no point in updating the ITRs until they need to send, in which case they will send Map-Requests to obtain a Map-Cache entry.

7. Routing Locator Reachability

This document defines several Control-Plane mechanisms for determining RLOC reachability. Please note that additional Data-Plane reachability mechanisms are defined in [I-D.ietf-lisp-rfc6830bis].

1. An ITR may receive an ICMP Network Unreachable or Host Unreachable message for an RLOC it is using. This indicates that the RLOC is likely down. Note that trusting ICMP messages may not be desirable, but neither is ignoring them completely. Implementations are encouraged to follow current best practices in treating these conditions [I-D.ietf-opsec-icmp-filtering].
2. When an ITR participates in the routing protocol that operates in the underlay routing system, it can determine that an RLOC is down when no Routing Information Base (RIB) entry exists that matches the RLOC IP address.
3. An ITR may receive an ICMP Port Unreachable message from a destination host. This occurs if an ITR attempts to use interworking [RFC6832] and LISP-encapsulated data is sent to a non-LISP-capable site.
4. An ITR may receive a Map-Reply from an ETR in response to a previously sent Map-Request. The RLOC source of the Map-Reply is likely up, since the ETR was able to send the Map-Reply to the ITR. Please note that in some scenarios the RLOC -from the outer header- can be a spoofable field.
5. An ITR/ETR pair can use the 'RLOC-Probing' mechanism described below.

When ITRs receive ICMP Network Unreachable or Host Unreachable messages as a method to determine unreachability, they will refrain from using Locators that are described in Locator lists of Map-Replies. However, using this approach is unreliable because many network operators turn off generation of ICMP Destination Unreachable messages.

If an ITR does receive an ICMP Network Unreachable or Host Unreachable message, it MAY originate its own ICMP Destination Unreachable message destined for the host that originated the data packet the ITR encapsulated.

This assumption does create a dependency: Locator unreachability is detected by the receipt of ICMP Host Unreachable messages. When a Locator has been determined to be unreachable, it is not used for

active traffic; this is the same as if it were listed in a Map-Reply with Priority 255.

The ITR can test the reachability of the unreachable Locator by sending periodic Map-Requests. Both Map-Requests and Map-Replies MUST be rate-limited, see Section 5.3 and Section 5.4 for information about rate-limiting. Locator reachability testing is never done with data packets, since that increases the risk of packet loss for end-to-end sessions.

7.1. RLOC-Probing Algorithm

RLOC-Probing is a method that an ITR or PITR can use to determine the reachability status of one or more Locators that it has cached in a Map-Cache entry. The probe-bit of the Map-Request and Map-Reply messages is used for RLOC-Probing.

RLOC-Probing is done in the control plane on a timer basis, where an ITR or PITR will originate a Map-Request destined to a locator address from one of its own locator addresses. A Map-Request used as an RLOC-probe is NOT encapsulated and NOT sent to a Map-Server or to the mapping database system as one would when requesting mapping data. The EID record encoded in the Map-Request is the EID-Prefix of the Map-Cache entry cached by the ITR or PITR. The ITR MAY include a mapping data record for its own database mapping information that contains the local EID-Prefixes and RLOCs for its site. RLOC-probes are sent periodically using a jittered timer interval.

When an ETR receives a Map-Request message with the probe-bit set, it returns a Map-Reply with the probe-bit set. The source address of the Map-Reply is set to the IP address of the outgoing interface the Map-Reply destination address routes to. The Map-Reply SHOULD contain mapping data for the EID-Prefix contained in the Map-Request. This provides the opportunity for the ITR or PITR that sent the RLOC-probe to get mapping updates if there were changes to the ETR's database mapping entries.

There are advantages and disadvantages of RLOC-Probing. The main benefit of RLOC-Probing is that it can handle many failure scenarios allowing the ITR to determine when the path to a specific Locator is reachable or has become unreachable, thus providing a robust mechanism for switching to using another Locator from the cached Locator. RLOC-Probing can also provide rough Round-Trip Time (RTT) estimates between a pair of Locators, which can be useful for network management purposes as well as for selecting low delay paths. The major disadvantage of RLOC-Probing is in the number of control messages required and the amount of bandwidth used to obtain those

benefits, especially if the requirement for failure detection times is very small.

8. Interactions with Other LISP Components

8.1. ITR EID-to-RLOC Mapping Resolution

An ITR is configured with one or more Map-Resolver addresses. These addresses are "Locators" (or RLOCs) and MUST be routable on the underlying core network; they MUST NOT need to be resolved through LISP EID-to-RLOC mapping, as that would introduce a circular dependency. When using a Map-Resolver, an ITR does not need to connect to any other database mapping system.

An ITR sends an Encapsulated Map-Request to a configured Map-Resolver when it needs an EID-to-RLOC mapping that is not found in its local Map-Cache. Using the Map-Resolver greatly reduces both the complexity of the ITR implementation and the costs associated with its operation.

In response to an Encapsulated Map-Request, the ITR can expect one of the following:

- o An immediate Negative Map-Reply (with action code of "Natively-Forward", 15-minute Time to Live (TTL)) from the Map-Resolver if the Map-Resolver can determine that the requested EID does not exist. The ITR saves the EID-Prefix returned in the Map-Reply in its cache, marks it as non-LISP-capable, and knows not to attempt LISP encapsulation for destinations matching it.
- o A Negative Map-Reply, with action code of "Natively-Forward", from a Map-Server that is authoritative (within the LISP deployment Section 1.1) for an EID-Prefix that matches the requested EID but that does not have an actively registered, more-specific EID-prefix. In this case, the requested EID is said to match a "hole" in the authoritative EID-Prefix. If the requested EID matches a more-specific EID-Prefix that has been delegated by the Map-Server but for which no ETRs are currently registered, a 1-minute TTL is returned. If the requested EID matches a non-delegated part of the authoritative EID-Prefix, then it is not a LISP EID and a 15-minute TTL is returned. See Section 8.2 for discussion of aggregate EID-Prefixes and details of Map-Server EID-Prefix matching.
- o A LISP Map-Reply from the ETR that owns the EID-to-RLOC mapping or possibly from a Map-Server answering on behalf of the ETR. See Section 8.4 for more details on Map-Resolver message processing.

Note that an ITR may be configured to both use a Map-Resolver and to participate in a LISP-ALT logical network. In such a situation, the ITR SHOULD send Map-Requests through the ALT network for any EID-Prefix learned via ALT BGP. Such a configuration is expected to be very rare, since there is little benefit to using a Map-Resolver if an ITR is already using LISP-ALT. There would be, for example, no need for such an ITR to send a Map-Request to a possibly non-existent EID (and rely on Negative Map-Replies) if it can consult the ALT database to verify that an EID-Prefix is present before sending that Map-Request.

8.2. EID-Prefix Configuration and ETR Registration

An ETR publishes its EID-Prefixes on a Map-Server by sending LISP Map-Register messages. A Map-Register message includes authentication data, so prior to sending a Map-Register message, the ETR and Map-Server MUST be configured with a pre-shared secret used to derive Map-Register authentication keys. A Map-Server's configuration SHOULD also include a list of the EID-Prefixes for which each ETR is authoritative. Upon receipt of a Map-Register from an ETR, a Map-Server accepts only EID-Prefixes that are configured for that ETR. Failure to implement such a check would leave the mapping system vulnerable to trivial EID-Prefix hijacking attacks.

In addition to the set of EID-Prefixes defined for each ETR that may register, a Map-Server is typically also configured with one or more aggregate prefixes that define the part of the EID numbering space assigned to it. When LISP-ALT is the database in use, aggregate EID-Prefixes are implemented as discard routes and advertised into ALT BGP. The existence of aggregate EID-Prefixes in a Map-Server's database means that it may receive Map Requests for EID-Prefixes that match an aggregate but do not match a registered prefix; Section 8.3 describes how this is handled.

Map-Register messages are sent periodically from an ETR to a Map-Server with a suggested interval between messages of one minute. A Map-Server SHOULD time out and remove an ETR's registration if it has not received a valid Map-Register message within the past three minutes. When first contacting a Map-Server after restart or changes to its EID-to-RLOC database mappings, an ETR MAY initially send Map-Register messages at an increased frequency, up to one every 20 seconds. This "quick registration" period is limited to five minutes in duration.

An ETR MAY request that a Map-Server explicitly acknowledge receipt and processing of a Map-Register message by setting the "want-map-notify" (M-bit) flag. A Map-Server that receives a Map-Register with this flag set will respond with a Map-Notify message. Typical use of

this flag by an ETR would be to set it for Map-Register messages sent during the initial "quick registration" with a Map-Server but then set it only occasionally during steady-state maintenance of its association with that Map-Server. Note that the Map-Notify message is sent to UDP destination port 4342, not to the source port specified in the original Map-Register message.

Note that a one-minute minimum registration interval during maintenance of an ETR-Map-Server association places a lower bound on how quickly and how frequently a mapping database entry can be updated. This may have implications for what sorts of mobility can be supported directly by the mapping system; shorter registration intervals or other mechanisms might be needed to support faster mobility in some cases. For a discussion on one way that faster mobility may be implemented for individual devices, please see [I-D.ietf-lisp-mn].

An ETR MAY also request, by setting the "proxy Map-Reply" flag (P-bit) in the Map-Register message, that a Map-Server answer Map-Requests instead of forwarding them to the ETR. See Section 7.1 for details on how the Map-Server sets certain flags (such as those indicating whether the message is authoritative and how returned Locators SHOULD be treated) when sending a Map-Reply on behalf of an ETR. When an ETR requests proxy reply service, it SHOULD include all RLOCs for all ETRs for the EID-Prefix being registered, along with the routable flag ("R-bit") setting for each RLOC. The Map-Server includes all of this information in Map-Reply messages that it sends on behalf of the ETR. This differs from a non-proxy registration, since the latter need only provide one or more RLOCs for a Map-Server to use for forwarding Map-Requests; the registration information is not used in Map-Replies, so it being incomplete is not incorrect.

An ETR that uses a Map-Server to publish its EID-to-RLOC mappings does not need to participate further in the mapping database protocol(s). When using a LISP-ALT mapping database, for example, this means that the ETR does not need to implement GRE or BGP, which greatly simplifies its configuration and reduces its cost of operation.

Note that use of a Map-Server does not preclude an ETR from also connecting to the mapping database (i.e., it could also connect to the LISP-ALT network), but doing so doesn't seem particularly useful, as the whole purpose of using a Map-Server is to avoid the complexity of the mapping database protocols.

8.3. Map-Server Processing

Once a Map-Server has EID-Prefixes registered by its client ETRs, it can accept and process Map-Requests for them.

In response to a Map-Request, the Map-Server first checks to see if the destination EID matches a configured EID-Prefix. If there is no match, the Map-Server returns a Negative Map-Reply with action code "Natively-Forward" and a 15-minute TTL. This can occur if a Map Request is received for a configured aggregate EID-Prefix for which no more-specific EID-Prefix exists; it indicates the presence of a non-LISP "hole" in the aggregate EID-Prefix.

Next, the Map-Server checks to see if any ETRs have registered the matching EID-Prefix. If none are found, then the Map-Server returns a Negative Map-Reply with action code "Natively-Forward" and a 1-minute TTL.

If the EID-prefix is either registered or not registered to the mapping system and there is a policy in the Map-Server to have the requestor drop packets for the matching EID-prefix, then a Drop/Policy-Denied action is returned. If the EID-prefix is registered or not registered and there is a authentication failure, then a Drop/Authentication-failure action is returned. If either of these actions result as a temporary state in policy or authentication then a Send-Map-Request action with 1-minute TTL MAY be returned to allow the requestor to retry the Map-Request.

If any of the registered ETRs for the EID-Prefix have requested proxy reply service, then the Map-Server answers the request instead of forwarding it. It returns a Map-Reply with the EID-Prefix, RLOCs, and other information learned through the registration process.

If none of the ETRs have requested proxy reply service, then the Map-Server re-encapsulates and forwards the resulting Encapsulated Map-Request to one of the registered ETRs. It does not otherwise alter the Map-Request, so any Map-Reply sent by the ETR is returned to the RLOC in the Map-Request, not to the Map-Server. Unless also acting as a Map-Resolver, a Map-Server should never receive Map-Replies; any such messages SHOULD be discarded without response, perhaps accompanied by the logging of a diagnostic message if the rate of Map-Replies is suggestive of malicious traffic.

8.4. Map-Resolver Processing

Upon receipt of an Encapsulated Map-Request, a Map-Resolver decapsulates the enclosed message and then searches for the requested EID in its local database of mapping entries (statically configured

or learned from associated ETRs if the Map-Resolver is also a Map-Server offering proxy reply service). If it finds a matching entry, it returns a LISP Map-Reply with the known mapping.

If the Map-Resolver does not have the mapping entry and if it can determine that the EID is not in the mapping database (for example, if LISP-ALT is used, the Map-Resolver will have an ALT forwarding table that covers the full EID space), it immediately returns a negative LISP Map-Reply, with action code "Natively-Forward" and a 15-minute TTL. To minimize the number of negative cache entries needed by an ITR, the Map-Resolver SHOULD return the least-specific prefix that both matches the original query and does not match any EID-Prefix known to exist in the LISP-capable infrastructure.

If the Map-Resolver does not have sufficient information to know whether the EID exists, it needs to forward the Map-Request to another device that has more information about the EID being requested. To do this, it forwards the unencapsulated Map-Request, with the original ITR RLOC as the source, to the mapping database system. Using LISP-ALT, the Map-Resolver is connected to the ALT network and sends the Map-Request to the next ALT hop learned from its ALT BGP neighbors. The Map-Resolver does not send any response to the ITR; since the source RLOC is that of the ITR, the ETR or Map-Server that receives the Map-Request over the ALT and responds will do so directly to the ITR.

8.4.1. Anycast Operation

A Map-Resolver can be set up to use "anycast", where the same address is assigned to multiple Map-Resolvers and is propagated through IGP routing, to facilitate the use of a topologically close Map-Resolver by each ITR.

ETRs MAY have anycast RLOC addresses which are registered as part of their RLOC-set to the mapping system. However, registrations MUST use their unique RLOC addresses, distinct authentication keys or different XTR-IDs to identify security associations with the Map-Servers.

9. Security Considerations

A LISP threat analysis can be found in [RFC7835]. In what follows we highlight security considerations that apply when LISP is deployed in environments such as those specified in Section 1.1, where the following assumptions hold:

1. The Mapping System is secure and trusted, and for the purpose of this security considerations the Mapping System is considered as one trusted element.
2. The ETRs have a pre-configured trust relationship with the Mapping System, which includes some form of shared secret, and the Mapping System is aware of which EIDs an ETR can advertise. How those keys and mappings gets established is out of the scope of this document.
3. LISP-SEC [I-D.ietf-lisp-sec] MUST be implemented. Network operators should carefully weight how the LISP-SEC threat model applies to their particular use case or deployment. If they decide to ignore a particular recommendation, they should make sure the risk associated with the corresponding threats is well understood.

The Map-Request/Map-Reply message exchange can be exploited by an attacker to mount DoS and/or amplification attacks. Attackers can send Map-Requests at high rates to overload LISP nodes and increase the state maintained by such nodes or consume CPU cycles. Such threats can be mitigated by systematically applying filters and rate limiters.

The Map-Request/Map-Reply message exchange can also be exploited to inject forged mappings directly in the ITR EID-to-RLOC map-cache. This can lead to traffic being redirected to the attacker, see further details in [RFC7835]. In addition, valid ETRs in the system can perform overclaiming attacks. In this case, attackers can claim to own an EID-prefix that is larger than the prefix owned by the ETR. Such attacks can be addressed by using LISP-SEC [I-D.ietf-lisp-sec]. The LISP-SEC protocol defines a mechanism for providing origin authentication, integrity protection, and prevention of 'man-in-the-middle' and 'prefix overclaiming' attacks on the Map-Request/Map-Reply exchange. In addition and while beyond the scope of securing an individual Map-Server or Map-Resolver, it should be noted that LISP-SEC can be complemented by additional security mechanisms defined by the Mapping System Infrastructure. For instance, BGP-based LISP-ALT [RFC6836] can take advantage of standards work on adding security to BGP while LISP-DDT [RFC8111] defines its own additional security mechanisms.

To publish an authoritative EID-to-RLOC mapping with a Map-Server using the Map-Register message, an ETR includes authentication data that is a MAC of the entire message using a key derived from the pre-shared secret. An implementation SHOULD support HMAC-SHA256-128+HKDF-SHA256 [RFC5869]. The Map-Register message includes protection for replay attacks by a man-in-the-middle. However, there

is a potential attack where a compromised ETR could overclaim the prefix it owns and successfully register it on its corresponding Map-Server. To mitigate this and as noted in Section 8.2, a Map-Server MUST verify that all EID-Prefixes registered by an ETR match the configuration stored on the Map-Server.

Deployments concerned about manipulations of Map-Request and Map-Reply messages, and malicious ETR EID prefix overclaiming MUST drop LISP Control Plane messages that do not contain LISP-SEC material (S-bit, EID-AD, OTK-AD, PKT-AD).

Mechanisms to encrypt, support privacy, prevent eavesdropping and packet tampering for messages exchanged between xTRs, xTRs and the mapping system, and nodes that make up the mapping system, SHOULD be deployed. Examples of this are DTLS [RFC6347] or LISP-crypto [RFC8061].

10. Privacy Considerations

As noted by [RFC6973] privacy is a complex issue that greatly depends on the specific protocol use-case and deployment. As noted in section 1.1 of [I-D.ietf-lisp-rfc6830bis] LISP focuses on use-cases where entities communicate over the public Internet while keeping separate addressing and topology. In what follows we detail the privacy threats introduced by the LISP Control Plane, the analysis is based on the guidelines detailed in [RFC6973].

LISP can use long-lived identifiers (EIDs) that survive mobility events. Such identifiers bind to the RLOCs of the nodes, which represents the topological location with respect to the specific LISP deployments. In addition, EID-to-RLOC mappings are typically considered public information within the LISP deployment when control-plane messages are not encrypted, and can be eavesdropped while Map-Request messages are sent to the corresponding Map-Resolvers or Map-Register messages to Map-Servers.

In this context, attackers can correlate the EID with the RLOC and track the corresponding user topological location and/or mobility. This can be achieved by off-path attackers, if they are authenticated, by querying the mapping system. Deployments concerned about this threat can use access control-lists or stronger authentication mechanisms [I-D.ietf-lisp-ecdsa-auth] in the mapping system to make sure that only authorized users can access this information (data minimization). Use of ephemeral EIDs [I-D.ietf-lisp-eid-anonymity] to achieve anonymity is another mechanism to lessen persistency and identity tracking.

11. Changes since RFC 6833

For implementation considerations, the following major changes have been made to this document since RFC 6833 was published:

- o A Map-Notify-Ack message is added in this document to provide reliability for Map-Notify messages. Any receiver of a Map-Notify message must respond with a Map-Notify-Ack message. Map-Servers who are senders of Map-Notify messages, must queue the Map-Notify contents until they receive a Map-Notify-Ack with the nonce used in the Map-Notify message. Note that implementations for Map-Notify-Ack support already exist and predate this document.
- o This document is incorporating the codepoint for the Map-Referral message from the LISP-DDT specification [RFC8111] to indicate that a Map-Server must send the final Map-Referral message when it participates in the LISP-DDT mapping system procedures.
- o The "L" and "D" bits are added to the Map-Request message. See Section 5.3 for details.
- o The "S", "I", "E", "T", "a", "R", and "M" bits are added to the Map-Register message. See Section 5.6 for details.
- o The 16-bit Key-ID field of the Map-Register message has been split into a 8-bit Key-ID field and a 8-bit Algorithm-ID field.
- o The nonce and the authentication data in the Map-Register message have a different behaviour, see Section 5.6 for details.
- o This document adds two new Action values that are in an EID-record that appear in Map-Reply, Map-Register, Map-Notify, and Map-Notify-Ack messages. The Drop/Policy-Denied and Drop/Auth-Failure are the descriptions for the two new action values. See Section 5.4 for details.

12. IANA Considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to this LISP Control-Plane specification, in accordance with BCP 26 [RFC8126].

There are three namespaces (listed in the sub-sections below) in LISP that have been registered.

- o LISP IANA registry allocations should not be made for purposes unrelated to LISP routing or transport protocols.

- o The following policies are used here with the meanings defined in BCP 26: "Specification Required", "IETF Review", "Experimental Use", and "First Come First Served".

12.1. LISP UDP Port Numbers

The IANA registry has allocated UDP port number 4342 for the LISP Control-Plane. IANA has updated the description for UDP port 4342 as follows:

Keyword	Port	Transport Layer	Description
-----	----	-----	-----
lisp-control	4342	udp	LISP Control Packets

12.2. LISP Packet Type Codes

It is being requested that the IANA be authoritative for LISP Packet Type definitions and it is requested to replace the [RFC6830] registry message references with the RFC number assigned to this document.

Based on deployment experience of [RFC6830], the Map-Notify-Ack message, message type 5, was added by this document. This document requests IANA to add it to the LISP Packet Type Registry.

Name	Number	Defined in
----	-----	-----
LISP Map-Notify-Ack	5	RFC6833bis

12.3. LISP Map-Reply EID-Record Action Codes

New ACT values can be allocated through IETF review or IESG approval. Four values have already been allocated by [RFC6830]. IANA is requested to replace the [RFC6830] reference for this registry with the RFC number assigned to this document and [RFC6830]. This specification changes the name of ACT type 3 value from "Drop" to "Drop/No-Reason" as well as adding two new ACT values, the "Drop/Policy-Denied" (type 4) and "Drop/Authentication-Failure" (type 5).

Value	Action	Description	Reference
4	Drop/Policy-Denied	A packet matching this Map-Cache entry is dropped because the target EWID is policy-denied by the xTR or the mapping system.	RFC6833bis
5	Drop/Auth-Failure	Packet matching the Map-Cache entry is dropped because the Map-Request for the target EID fails an authentication check by the xTR or the mapping system.	RFC6833bis

LISP Map-Reply Action Values

In addition, LISP has a number of flag fields and reserved fields, such as the LISP header flags field [I-D.ietf-lisp-rfc6830bis]. New bits for flags in these fields can be implemented after IETF review or IESG approval, but these need not be managed by IANA.

12.4. LISP Address Type Codes

LISP Canonical Address Format (LCAF) [RFC8060] is an 8-bit field that defines LISP-specific encodings for AFI value 16387. LCAF encodings are used for specific use-cases where different address types for EID-records and RLOC-records are required.

The IANA registry "LISP Canonical Address Format (LCAF) Types" is used for LCAF types. The registry for LCAF types use the Specification Required policy [RFC8126]. Initial values for the registry as well as further information can be found in [RFC8060].

Therefore, there is no longer a need for the "LISP Address Type Codes" registry requested by [RFC6830]. This document requests to remove it.

12.5. LISP Algorithm ID Numbers

In [RFC6830], a request for a "LISP Key ID Numbers" registry was submitted. This document renames the registry to "LISP Algorithm ID Numbers" and requests the IANA to make the name change.

The following Algorithm ID values are defined by this specification as used in any packet type that references a 'Algorithm ID' field:

Name	Number	MAC	KDF	
None	0	None	None	
HMAC-SHA-1-96-None	1	[RFC2404]	None	
HMAC-SHA-256-128-None		2	[RFC4868]	None
HMAC-SHA256-128+HKDF-SHA256	3	[RFC4868]	[RFC4868]	

Number values are in the range of 0 to 255. The allocation of values is on a first come first served basis.

12.6. LISP Bit Flags

This document asks IANA to create a registry for allocation of bits in several headers of the LISP control plane, namely in the Map-Request, Map-Reply, Map-Register, Encapsulated Control Message (ECM) messages. Bit allocations are also requested for EID-records and RLOC-records. The registry created should be named "LISP Control Plane Header Bits". A sub-registry needs to be created per each message and EID-record. The name of each sub-registry is indicated below, along with its format and allocation of bits defined in this document. Any additional bits allocation, requires a specification, according with [RFC8126] policies.

Sub-Registry: Map-Request Header Bits [Section 5.2]:

0										1										2										3																																																	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1																																																
Type=1										A M P S p s R R										Rsvd										L D										IRC																				Record Count																			

Spec Name	IANA Name	Bit Position	Description
A	map-request-A	4	Authoritative Bit
M	map-request-M	5	Map Data Present Bit
P	map-request-P	6	RLOC-Probe Request Bit
S	map-request-S	7	Solicit Map-Request (SMR) Bit
p	map-request-p	8	Proxy-ITR Bit
s	map-request-s	9	Solicit Map-Request Invoked Bit
L	map-request-L	17	Local xTR Bit
D	map-request-D	18	Don't Map-Reply Bit

LISP Map-Request Header Bits

Sub-Registry: Map-Reply Header Bits [Section 5.4]:

0	1	2	3
0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9	0 1
Type=2 P E S	Reserved	Record Count	

Spec Name	IANA Name	Bit Position	Description
P	map-reply-P	4	RLOC-Probe Bit
E	map-reply-E	5	Echo Nonce Capable Bit
S	map-reply-S	6	Security Bit

LISP Map-Reply Header Bits

Sub-Registry: Map-Register Header Bits [Section 5.6]:

0	1	2	3
0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9	0 1
Type=3 P S I	Reserved	E T a R M	Record Count

Spec Name	IANA Name	Bit Position	Description
P	map-register-P	4	Proxy Map-Reply Bit
S	map-register-S	5	LISP-SEC Capable Bit
I	map-register-I	6	xTR-ID present flag

LISP Map-Register Header Bits

Sub-Registry: Encapsulated Control Message (ECM) Header Bits
[Section 5.8]:

0	1	2	3
0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9	0 1
Type=8	S D E M	Reserved	

Spec Name	IANA Name	Bit Position	Description
S	ecm-S	4	Security Bit
D	ecm-D	5	LISP-DDT Bit
E	ecm-E	6	Forward to ETR Bit
M	ecm-M	7	Destined to Map-Server Bit

LISP Encapsulated Control Message (ECM) Header Bits

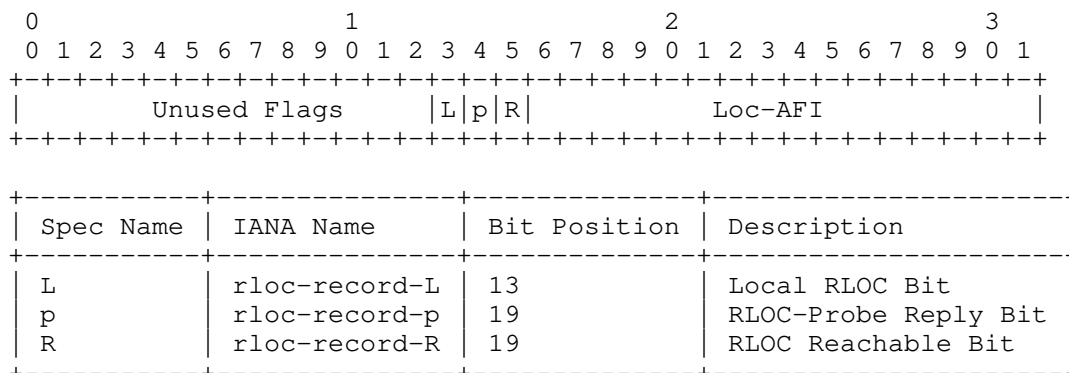
Sub-Registry: EID-Record Header Bits [Section 5.4]:

0	1	2	3
0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9	0 1
Locator Count	EID mask-len	ACT A	Reserved

Spec Name	IANA Name	Bit Position	Description
A	eid-record-A	19	Authoritative Bit

LISP EID-Record Header Bits

Sub-Registry: RLOC-Record Header Bits [Section 5.4]:



LISP RLOC-Record Header Bits

13. References

13.1. Normative References

- [I-D.ietf-lisp-6834bis]
 Iannone, L., Saucez, D., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Map-Versioning", draft-ietf-lisp-6834bis-09 (work in progress), August 2021.
- [I-D.ietf-lisp-rfc6830bis]
 Farinacci, D., Fuller, V., Meyer, D., Lewis, D., and A. Cabellos, "The Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-rfc6830bis-36 (work in progress), November 2020.
- [I-D.ietf-lisp-rfc8113bis]
 Boucadair, M. and C. Jacquenet, "Locator/ID Separation Protocol (LISP): Shared Extension Message & IANA Registry for Packet Type Allocations", draft-ietf-lisp-rfc8113bis-03 (work in progress), January 2019.
- [I-D.ietf-lisp-sec]
 Maino, F., Ermagan, V., Cabellos, A., and D. Saucez, "LISP-Security (LISP-SEC)", draft-ietf-lisp-sec-25 (work in progress), December 2021.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC2404] Madson, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, DOI 10.17487/RFC2404, November 1998, <<https://www.rfc-editor.org/info/rfc2404>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, DOI 10.17487/RFC4868, May 2007, <<https://www.rfc-editor.org/info/rfc4868>>.
- [RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <<https://www.rfc-editor.org/info/rfc5869>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

13.2. Informative References

- [AFI] "Address Family Identifier (AFIs)", ADDRESS FAMILY NUMBERS <http://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml?>, Febuary 2007.
- [GTP-3GPP] "General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)", TS.29.281 <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1699>, January 2015.

- [I-D.herbert-intarea-ila]
Herbert, T. and P. Lapukhov, "Identifier-locator addressing for IPv6", draft-herbert-intarea-ila-01 (work in progress), March 2018.
- [I-D.ietf-lisp-ecdsa-auth]
Farinacci, D. and E. Nordmark, "LISP Control-Plane ECDSA Authentication and Authorization", draft-ietf-lisp-ecdsa-auth-07 (work in progress), February 2022.
- [I-D.ietf-lisp-eid-anonymity]
Farinacci, D., Pillay-Esnault, P., and W. Haddad, "LISP EID Anonymity", draft-ietf-lisp-eid-anonymity-12 (work in progress), March 2022.
- [I-D.ietf-lisp-eid-mobility]
Comeras, M. P., Ashtaputre, V., Maino, F., Moreno, V., and D. Farinacci, "LISP L2/L3 EID Mobility Using a Unified Control Plane", draft-ietf-lisp-eid-mobility-09 (work in progress), January 2022.
- [I-D.ietf-lisp-gpe]
Maino, F., Lemon, J., Agarwal, P., Lewis, D., and M. Smith, "LISP Generic Protocol Extension", draft-ietf-lisp-gpe-19 (work in progress), July 2020.
- [I-D.ietf-lisp-introduction]
Cabellos, A. and D. S. (Ed.), "An Architectural Introduction to the Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-introduction-15 (work in progress), September 2021.
- [I-D.ietf-lisp-mn]
Farinacci, D., Lewis, D., Meyer, D., and C. White, "LISP Mobile Node", draft-ietf-lisp-mn-11 (work in progress), January 2022.
- [I-D.ietf-lisp-pubsub]
Rodriguez-Natal, A., Ermagan, V., Cabellos, A., Barkai, S., and M. Boucadair, "Publish/Subscribe Functionality for LISP", draft-ietf-lisp-pubsub-09 (work in progress), June 2021.
- [I-D.ietf-nvo3-vxlan-gpe]
(Editor), F. M., (editor), L. K., and U. E. (editor), "Generic Protocol Extension for VXLAN (VXLAN-GPE)", draft-ietf-nvo3-vxlan-gpe-12 (work in progress), September 2021.

- [I-D.ietf-opsec-icmp-filtering]
Gont, F., Gont, G., and C. Pignataro, "Recommendations for filtering ICMP messages", draft-ietf-opsec-icmp-filtering-04 (work in progress), July 2013.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC1071] Braden, R., Borman, D., and C. Partridge, "Computing the Internet checksum", RFC 1071, DOI 10.17487/RFC1071, September 1988, <<https://www.rfc-editor.org/info/rfc1071>>.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <<https://www.rfc-editor.org/info/rfc2104>>.
- [RFC2890] Dommetty, G., "Key and Sequence Number Extensions to GRE", RFC 2890, DOI 10.17487/RFC2890, September 2000, <<https://www.rfc-editor.org/info/rfc2890>>.
- [RFC4984] Meyer, D., Ed., Zhang, L., Ed., and K. Fall, Ed., "Report from the IAB Workshop on Routing and Addressing", RFC 4984, DOI 10.17487/RFC4984, September 2007, <<https://www.rfc-editor.org/info/rfc4984>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC6831] Farinacci, D., Meyer, D., Zwiebel, J., and S. Venaas, "The Locator/ID Separation Protocol (LISP) for Multicast Environments", RFC 6831, DOI 10.17487/RFC6831, January 2013, <<https://www.rfc-editor.org/info/rfc6831>>.

- [RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <<https://www.rfc-editor.org/info/rfc6832>>.
- [RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, DOI 10.17487/RFC6836, January 2013, <<https://www.rfc-editor.org/info/rfc6836>>.
- [RFC6837] Lear, E., "NERD: A Not-so-novel Endpoint ID (EID) to Routing Locator (RLOC) Database", RFC 6837, DOI 10.17487/RFC6837, January 2013, <<https://www.rfc-editor.org/info/rfc6837>>.
- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <<https://www.rfc-editor.org/info/rfc6973>>.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC7835] Saucez, D., Iannone, L., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Threat Analysis", RFC 7835, DOI 10.17487/RFC7835, April 2016, <<https://www.rfc-editor.org/info/rfc7835>>.
- [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.
- [RFC8061] Farinacci, D. and B. Weis, "Locator/ID Separation Protocol (LISP) Data-Plane Confidentiality", RFC 8061, DOI 10.17487/RFC8061, February 2017, <<https://www.rfc-editor.org/info/rfc8061>>.
- [RFC8111] Fuller, V., Lewis, D., Ermagan, V., Jain, A., and A. Smirnov, "Locator/ID Separation Protocol Delegated Database Tree (LISP-DDT)", RFC 8111, DOI 10.17487/RFC8111, May 2017, <<https://www.rfc-editor.org/info/rfc8111>>.

- [RFC8378] Moreno, V. and D. Farinacci, "Signal-Free Locator/ID Separation Protocol (LISP) Multicast", RFC 8378, DOI 10.17487/RFC8378, May 2018, <<https://www.rfc-editor.org/info/rfc8378>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

Appendix A. Acknowledgments

The original authors would like to thank Greg Schudel, Darrel Lewis, John Zwiebel, Andrew Partan, Dave Meyer, Isidor Kouvelas, Jesper Skriver, Fabio Maino, and members of the `lisp@ietf.org` mailing list for their feedback and helpful suggestions.

Special thanks are due to Noel Chiappa for his extensive work and thought about caching in Map-Resolvers.

The current authors would like to give a sincere thank you to the people who help put LISP on standards track in the IETF. They include Joel Halpern, Luigi Iannone, Deborah Brungard, Fabio Maino, Scott Bradner, Kyle Rose, Takeshi Takahashi, Sarah Banks, Pete Resnick, Colin Perkins, Mirja Kuhlewind, Francis Dupont, Benjamin Kaduk, Eric Rescorla, Alvaro Retana, Alexey Melnikov, Alissa Cooper, Suresh Krishnan, Alberto Rodriguez-Natal, Vina Ermagen, Mohamed Boucadair, Brian Trammell, Sabrina Tanamal, and John Drake. The contributions they offered greatly added to the security, scale, and robustness of the LISP architecture and protocols.

Appendix B. Document Change Log

[RFC Editor: Please delete this section on publication as RFC.]

B.1. Changes to draft-ietf-lisp-rfc6833bis-31

- o Posted May 2020.
- o Added reference to 6834bis when describing the syntax and action of the Map-Version field in the Map-Reply section. This is so we can advance the Map-Versioning draft rfc6834bis to proposed standard.

B.2. Changes to draft-ietf-lisp-rfc6833bis-26

- o Posted November 2019.
- o Fixed the required (MUST implement) authentication algorithms.
- o Fixed a large set of minor comments and edits.

B.3. Changes to draft-ietf-lisp-rfc6833bis-25

- o Posted June 2019.
- o Added change requested by Mirja describing Record Count in an EID-record.

- o Fixed Requirements Notation section per Pete.
 - o Added KDF for shared-secret
 - o Specified several rate-limiters for control messages
- B.4. Changes to draft-ietf-lisp-rfc6833bis-24
- o Posted February 2019.
 - o Added suggested text from Albert that Benjamin Kaduk agreed with.
 - o Added suggested editorial comments from Alvaro's review.
 - o Ran document through IDnits. Fixed bugs found.
- B.5. Changes to draft-ietf-lisp-rfc6833bis-23
- o Posted December 2018.
 - o Added to Security Considerations section that deployments that care about prefix over claiming should use LISP-SEC.
 - o Added to Security Considerations section that DTLS or LISP-crypto be used for control-plane privacy.
 - o Make LISP-SEC a normative reference.
 - o Make it more clear where field descriptions are specified when referencing to the same fields in other packet types.
- B.6. Changes to draft-ietf-lisp-rfc6833bis-22
- o Posted week after IETF November 2018.
 - o No longer need to use IPSEC for replay attacks.
- B.7. Changes to draft-ietf-lisp-rfc6833bis-21
- o Posted early November 2018.
 - o Added I-bit back in because its necessary to use for Map-Register replay attack scenarios. The Map-Server tracks the nonce per xTR-ID to detect duplicate or replayed Map-Register messages.

- B.8. Changes to draft-ietf-lisp-rfc6833bis-20
- o Posted late October 2018.
 - o Changed description about "reserved" bits to state "reserved and unassigned".
 - o Make it more clear how Map-Register nonce processing is performed in an ETR and Map-Server.
- B.9. Changes to draft-ietf-lisp-rfc6833bis-19
- o Posted mid October 2018.
 - o Added Fabio text to the Security Considerations section.
- B.10. Changes to draft-ietf-lisp-rfc6833bis-18
- o Posted mid October 2018.
 - o Fixed comments from Eric after more email clarity.
- B.11. Changes to draft-ietf-lisp-rfc6833bis-17
- o Posted early October 2018.
 - o Changes to reflect comments from Sep 27th Telechat.
 - o Added all flag bit definitions as request for allocation in IANA Considerations section.
 - o Added an applicability statement in section 1 to address security concerns from Telechat.
 - o Moved m-bit description and IANA request to draft-ietf-lisp-mn.
 - o Moved I-bit description and IANA request to draft-ietf-lisp-pubsub.
- B.12. Changes to draft-ietf-lisp-rfc6833bis-16
- o Posted Late-September 2018.
 - o Re-wrote Security Considerations section. Thanks Albert.
 - o Added Alvaro text to be more clear about IANA actions.

- B.13. Changes to draft-ietf-lisp-rfc6833bis-15
- o Posted mid-September 2018.
 - o Changes to reflect comments from Colin and Mirja.
- B.14. Changes to draft-ietf-lisp-rfc6833bis-14
- o Posted September 2018.
 - o Changes to reflect comments from Genart, RTGarea, and Secdir reviews.
- B.15. Changes to draft-ietf-lisp-rfc6833bis-13
- o Posted August 2018.
 - o Final editorial changes before RFC submission for Proposed Standard.
 - o Added section "Changes since RFC 6833" so implementators are informed of any changes since the last RFC publication.
- B.16. Changes to draft-ietf-lisp-rfc6833bis-12
- o Posted late July 2018.
 - o Moved RFC6830bis and RFC6834bis to Normative References.
- B.17. Changes to draft-ietf-lisp-rfc6833bis-11
- o Posted July 2018.
 - o Fixed Luigi editorial comments to ready draft for RFC status and ran through IDNITs again.
- B.18. Changes to draft-ietf-lisp-rfc6833bis-10
- o Posted after LISP WG at IETF week March.
 - o Move AD field encoding after S-bit in the ECM packet format description section.
 - o Say more about when the new Drop actions should be sent.

B.19. Changes to draft-ietf-lisp-rfc6833bis-09

- o Posted March IETF week 2018.
- o Fixed editorial comments submitted by document shepherd Luigi Iannone.

B.20. Changes to draft-ietf-lisp-rfc6833bis-08

- o Posted March 2018.
- o Added RLOC-probing algorithm.
- o Added Solicit-Map Request algorithm.
- o Added several mechanisms (from 6830bis) regarding Routing Locator Reachability.
- o Added port 4342 to IANA Considerations section.

B.21. Changes to draft-ietf-lisp-rfc6833bis-07

- o Posted December 2017.
- o Make it more clear in a couple of places that RLOCs are used to locate ETRs more so than for Map-Server Map-Request forwarding.
- o Make it clear that "encapsualted" for a control message is an ECM based message.
- o Make it more clear what messages use source-port 4342 and which ones use destinatio-port 4342.
- o Don't make DDT references when the mapping transport system can be of any type and the referneced text is general to it.
- o Generalize text when referring to the format of an EID-prefix. Can use othe AFIs then IPv4 and IPv6.
- o Many editorial changes to clarify text.
- o Changed some "must", "should", and "may" to capitalized.
- o Added definitions for Map-Request and Map-Reply messages.
- o Ran document through IDNITs.

B.22. Changes to draft-ietf-lisp-rfc6833bis-06

- o Posted October 2017.
- o Spec the I-bit to include the xTR-ID in a Map-Request message to be consistent with the Map-Register message and to anticipate the introduction of pubsub functionality to allow Map-Requests to subscribe to RLOC-set changes.
- o Updated references for individual submissions that became working group documents.
- o Updated references for working group documents that became RFCs.

B.23. Changes to draft-ietf-lisp-rfc6833bis-05

- o Posted May 2017.
- o Update IANA Considerations section based on new requests from this document and changes from what was requested in [RFC6830].

B.24. Changes to draft-ietf-lisp-rfc6833bis-04

- o Posted May 2017.
- o Clarify how the Key-ID field is used in Map-Register and Map-Notify messages. Break the 16-bit field into a 8-bit Key-ID field and a 8-bit Algorithm-ID field.
- o Move the Control-Plane codepoints from the IANA Considerations section of RFC6830bis to the IANA Considerations section of this document.
- o In the "LISP Control Packet Type Allocations" section, indicate how message Types are IANA allocated and how experimental RFC8113 sub-types should be requested.

B.25. Changes to draft-ietf-lisp-rfc6833bis-03

- o Posted April 2017.
- o Add types 9-14 and specify they are not assigned.
- o Add the "LISP Shared Extension Message" type and point to RFC8113.

B.26. Changes to draft-ietf-lisp-rfc6833bis-02

- o Posted April 2017.
- o Clarify that the LISP Control-Plane document defines how the LISP Data-Plane uses Map-Requests with either the SMR-bit set or the P-bit set supporting mapping updates and RLOC-probing. Indicating that other Data-Planes can use the same mechanisms or their own defined mechanisms to achieve the same functionality.

B.27. Changes to draft-ietf-lisp-rfc6833bis-01

- o Posted March 2017.
- o Include references to new RFCs published.
- o Remove references to self.
- o Change references from RFC6830 to RFC6830bis.
- o Add two new action/reasons to a Map-Reply has posted to the LISP WG mailing list.
- o In intro section, add refernece to I-D.ietf-lisp-introduction.
- o Removed Open Issues section and references to "experimental".

B.28. Changes to draft-ietf-lisp-rfc6833bis-00

- o Posted December 2016.
- o Created working group document from draft-farinacci-lisp-rfc6833-00 individual submission. No other changes made.

B.29. Changes to draft-farinacci-lisp-rfc6833bis-00

- o Posted November 2016.
- o This is the initial draft to turn RFC 6833 into RFC 6833bis.
- o The document name has changed from the "Locator/ID Separation Protocol (LISP) Map-Server Interface" to the "Locator/ID Separation Protocol (LISP) Control-Plane".
- o The fundamental change was to move the Control-Plane messages from RFC 6830 to this document in an effort so any IETF developed or industry created Data-Plane could use the LISP mapping system and Control-Plane.

- o Update Control-Plane messages to incorporate what has been implemented in products during the early phase of LISP development but wasn't able to make it into RFC6830 and RFC6833 to make the Experimental RFC deadline.
- o Indicate there may be nodes in the mapping system that are not MRs or MSs, that is a ALT-node or a DDT-node.
- o Include LISP-DDT in Map-Resolver section and explain how they maintain a referral-cache.
- o Removed open issue about additional state in Map-Servers. With [RFC8111], Map-Servers have the same registration state and can give Map-Resolvers complete information in ms-ack Map-Referral messages.
- o Make reference to the LISP Threats Analysis RFC [RFC7835].

Authors' Addresses

Dino Farinacci
lispers.net

EEmail: farinacci@gmail.com

Fabio Maino
Cisco Systems

EEmail: fmaino@cisco.com

Vince Fuller
vaf.net Internet Consulting

EEmail: vaf@vaf.net

Albert Cabellos
UPC/BarcelonaTech
Campus Nord, C. Jordi Girona 1-3
Barcelona, Catalunya
Spain

EEmail: acabello@ac.upc.edu

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 11 June 2022

F.M. Maino
Cisco Systems
V.E. Ermagan
Google
A.C. Cabellos
Universitat Politecnica de Catalunya
D.S. Saucez
Inria
8 December 2021

LISP-Security (LISP-SEC)
draft-ietf-lisp-sec-25

Abstract

This memo specifies LISP-SEC, a set of security mechanisms that provides origin authentication, integrity and anti-replay protection to LISP's EID-to-RLOC mapping data conveyed via mapping lookup process. LISP-SEC also enables verification of authorization on EID-prefix claims in Map-Reply messages.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 11 June 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights

and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
2. Requirements Notation	3
3. Definition of Terms	3
4. LISP-SEC Threat Model	4
5. Protocol Operations	5
6. LISP-SEC Control Messages Details	7
6.1. Encapsulated Control Message LISP-SEC Extensions	7
6.2. Map-Reply LISP-SEC Extensions	10
6.3. Map-Register LISP-SEC Extensions	11
6.4. ITR Processing: Generating a Map-Request	12
6.4.1. PITR Processing	12
6.5. Encrypting and Decrypting an OTK	13
6.5.1. Unencrypted OTK	14
6.6. Map-Resolver Processing	15
6.7. Map-Server Processing	15
6.7.1. Generating a LISP-SEC Protected Encapsulated Map-Request	17
6.7.2. Generating a Proxy Map-Reply	18
6.8. ETR Processing	18
6.9. ITR Processing: Receiving a Map-Reply	18
6.9.1. Map-Reply Record Validation	20
7. Security Considerations	21
7.1. Mapping System Security	21
7.2. Random Number Generation	21
7.3. Map-Server and ETR Colocation	21
7.4. Deploying LISP-SEC	22
7.5. Shared Keys Provisioning	22
7.6. Replay Attacks	22
7.7. Message Privacy	23
7.8. Denial of Service and Distributed Denial of Service Attacks	23
8. IANA Considerations	23
8.1. ECM AD Type Registry	23
8.2. Map-Reply AD Type Registry	23
8.3. HMAC Functions	24
8.4. Key Wrap Functions	24
8.5. Key Derivation Functions	24
9. Acknowledgements	25
10. References	25
10.1. Normative References	25
10.2. Informational References	26

Authors' Addresses	27
--------------------	----

1. Introduction

The Locator/ID Separation Protocol [I-D.ietf-lisp-rfc6830bis], [I-D.ietf-lisp-rfc6833bis] is a network-layer-based protocol that enables separation of IP addresses into two new numbering spaces: Endpoint Identifiers (EIDs) and Routing Locators (RLOCs). EID-to-RLOC mappings are stored in a database, the LISP Mapping System, and made available via the Map-Request/Map-Reply lookup process. If these EID-to-RLOC mappings, carried through Map-Reply messages, are transmitted without integrity protection, an adversary can manipulate them and hijack the communication, impersonate the requested EID, or mount Denial of Service or Distributed Denial of Service attacks. Also, if the Map-Reply message is transported unauthenticated, an adversarial LISP entity can overclaim an EID-prefix and maliciously redirect traffic directed to a large number of hosts. The LISP-SEC threat model, described in Section 4, is built on top of the LISP threat model defined in [RFC7835], that includes a detailed description of "overclaiming" attack.

This memo specifies LISP-SEC, a set of security mechanisms that provides origin authentication, integrity and anti-replay protection to LISP's EID-to-RLOC mapping data conveyed via mapping lookup process. LISP-SEC also enables verification of authorization on EID-prefix claims in Map-Reply messages, ensuring that the sender of a Map-Reply that provides the location for a given EID-prefix is entitled to do so according to the EID prefix registered in the associated Map-Server. Map-Register/Map-Notify security, including the right for a LISP entity to register an EID-prefix or to claim presence at an RLOC, is out of the scope of LISP-SEC as those protocols are protected by the security mechanisms specified in [I-D.ietf-lisp-rfc6833bis]. However, LISP-SEC extends the Map-Register message to allow an ITR to securely downgrade to non LISP-SEC Map-Requests. Additional security considerations are described in Section 6.

2. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Definition of Terms

One-Time Key (OTK): An ephemeral randomly generated key that must be used for a single Map-Request/Map-Reply exchange.

ITR One-Time Key (ITR-OTK): The One-Time Key generated at the Ingress Tunnel Router (ITR).

MS One-Time Key (MS-OTK): The One-Time Key generated at the Map-Server.

Authentication Data (AD): Metadata that is included either in a LISP Encapsulated Control Message (ECM) header, as defined in [I-D.ietf-lisp-rfc6833bis], or in a Map-Reply message to support confidentiality, integrity protection, and verification of EID-prefix authorization.

OTK Authentication Data (OTK-AD): The portion of ECM Authentication Data that contains a One-Time Key.

EID Authentication Data (EID-AD): The portion of ECM and Map-Reply Authentication Data used for verification of EID-prefix authorization.

Packet Authentication Data (PKT-AD): The portion of Map-Reply Authentication Data used to protect the integrity of the Map-Reply message.

For definitions of other terms, notably Map-Request, Map-Reply, Ingress Tunnel Router (ITR), Egress Tunnel Router (ETR), Map-Server (MS), and Map-Resolver (MR) please consult the LISP specification [I-D.ietf-lisp-rfc6833bis].

4. LISP-SEC Threat Model

LISP-SEC addresses the control plane threats, described in section 3.7 and 3.8 of [RFC7835], that target EID-to-RLOC mappings, including manipulations of Map-Request and Map-Reply messages, and malicious ETR EID prefix overclaiming. LISP-SEC makes two main assumptions: (1) the LISP mapping system is expected to deliver a Map-Request message to their intended destination ETR as identified by the EID, and (2) no man-in-the-middle (MITM) attack can be mounted within the LISP Mapping System. How the Mapping System is protected from MITM attacks depends from the particular Mapping System used, and is out of the scope of this memo. Furthermore, while LISP-SEC enables detection of EID prefix overclaiming attacks, it assumes that Map-Servers can verify the EID prefix authorization at registration time.

According to the threat model described in [RFC7835] LISP-SEC assumes that any kind of attack, including MITM attacks, can be mounted outside of the boundaries of the LISP mapping system. An on-path attacker, outside of the LISP mapping system can, for example, hijack Map-Request and Map-Reply messages, spoofing the identity of a LISP node. Another example of on-path attack, called overclaiming attack, can be mounted by a malicious Egress Tunnel Router (ETR), by overclaiming the EID-prefixes for which it is authoritative. In this way the ETR can maliciously redirect traffic directed to a large number of hosts.

5. Protocol Operations

The goal of the security mechanisms defined in [I-D.ietf-lisp-rfc6833bis] is to prevent unauthorized insertion of mapping data by providing origin authentication and integrity protection for the Map-Register, and by using the nonce to detect unsolicited Map-Reply sent by off-path attackers.

LISP-SEC builds on top of the security mechanisms defined in [I-D.ietf-lisp-rfc6833bis] to address the threats described in Section 4 by leveraging the trust relationships existing among the LISP entities participating to the exchange of the Map-Request/Map-Reply messages. Those trust relationships are used to securely distribute, as described in Section 8.4, a per-message One-Time Key (OTK) that provides origin authentication, integrity and anti-replay protection to mapping data conveyed via the mapping lookup process, and that effectively prevent overclaiming attacks. The processing of security parameters during the Map-Request/Map-Reply exchange is as follows:

- * Per each Map-Request message a new ITR-OTK is generated and stored at the ITR, and securely transported to the Map-Server.
- * The Map-Server uses the ITR-OTK to compute a Keyed-Hashing for Message Authentication (HMAC) [RFC2104] that protects the integrity of the mapping data known to the Map-Server to prevent overclaiming attacks. The Map-Server also derives a new OTK, the MS-OTK, that is passed to the ETR, by applying a Key Derivation Function (KDF) (e.g. [RFC5869]) to the ITR-OTK.
- * The ETR uses the MS-OTK to compute an HMAC that protects the integrity of the Map-Reply sent to the ITR.
- * Finally, the ITR uses the stored ITR-OTK to verify the integrity of the mapping data provided by both the Map-Server and the ETR, and to verify that no overclaiming attacks were mounted along the path between the Map-Server and the ITR.

Section 6 provides the detailed description of the LISP-SEC control messages and their processing, while the rest of this section describes the flow of LISP protocol operations at each entity involved in the Map-Request/Map-Reply exchange:

1. The ITR, upon needing to transmit a Map-Request message, generates and stores an OTK (ITR-OTK). This ITR-OTK is included into the Encapsulated Control Message (ECM) that contains the Map-Request sent to the Map-Resolver. ITR-OTK confidentiality and integrity protection MUST be provided in the path between the ITR and the Map-Resolver. This can be achieved either by encrypting the ITR-OTK with the pre-shared secret known to the ITR and the Map-Resolver (as specified in Section 6.5), or by enabling DTLS between the ITR and the Map-Resolver.
2. The Map-Resolver decapsulates the ECM message, decrypts the ITR-OTK, if needed, and forwards through the Mapping System the received Map-Request and the ITR-OTK, as part of a new ECM message. The LISP Mapping System delivers the ECM to the appropriate Map-Server, as identified by the EID destination address of the Map-Request. As mentioned in Section 4, how the Mapping System is protected from MITM attacks depends from the particular Mapping Systems used, and is out of the scope of this memo.
3. The Map-Server is configured with the location mappings and policy information for the ETR responsible for the EID destination address. Using this preconfigured information, the Map-Server, after the decapsulation of the ECM message, finds the longest match EID-prefix that covers the requested EID in the received Map-Request. The Map-Server adds this EID-prefix, together with an HMAC computed using the ITR-OTK, to a new Encapsulated Control Message that contains the received Map-Request.
4. The Map-Server derives a new OTK, the MS-OTK, by applying a Key Derivation Function (KDF) to the ITR-OTK. This MS-OTK is included in the Encapsulated Control Message that the Map-Server uses to forward the Map-Request to the ETR. MS-OTK confidentiality and integrity protection MUST be provided in the path between the Map-Server and the ETR. This can be achieved either by encrypting the MS-OTK with the pre-shared secret known to the Map-Server and the ETR (as specified in Section 6.5), or by enabling DTLS between the Map-Server and the ETR.

5. If the Map-Server is acting in proxy mode, as specified in [I-D.ietf-lisp-rfc6833bis], the ETR is not involved in the generation of the Map-Reply and steps 6 and 7 are skipped. In this case the Map-Server generates the Map-Reply on behalf of the ETR as described in Section 6.7.2.
6. The ETR, upon receiving the ECM encapsulated Map-Request from the Map-Server, decrypts the MS-OTK, if needed, and originates a Map-Reply that contains the EID-to-RLOC mapping information as specified in [I-D.ietf-lisp-rfc6833bis].
7. The ETR computes an HMAC over the Map-Reply, keyed with MS-OTK to protect the integrity of the whole Map-Reply. The ETR also copies the EID-prefix authorization data that the Map-Server included in the ECM encapsulated Map-Request into the Map-Reply message. The ETR then sends the complete Map-Reply message to the requesting ITR.
8. The ITR, upon receiving the Map-Reply, uses the locally stored ITR-OTK to verify the integrity of the EID-prefix authorization data included in the Map-Reply by the Map-Server. The ITR computes the MS-OTK by applying the same KDF (as specified in the ECM encapsulated Map-Reply) used by the Map-Server, and verifies the integrity of the Map-Reply. If the integrity checks fail, the Map-Reply MUST be discarded. Also, if the EID-prefixes claimed by the ETR in the Map-Reply are not equal or more specific than the EID-prefix authorization data inserted by the Map-Server, the ITR MUST discard the Map-Reply.

6. LISP-SEC Control Messages Details

LISP-SEC metadata associated with a Map-Request is transported within the Encapsulated Control Message that contains the Map-Request.

LISP-SEC metadata associated with the Map-Reply is transported within the Map-Reply itself.

6.1. Encapsulated Control Message LISP-SEC Extensions

LISP-SEC uses the ECM defined in [I-D.ietf-lisp-rfc6833bis] with S bit set to 1 to indicate that the LISP header includes Authentication Data (AD). The format of the LISP-SEC ECM Authentication Data is defined in Figure 1. OTK-AD stands for One-Time Key Authentication Data and EID-AD stands for EID Authentication Data.

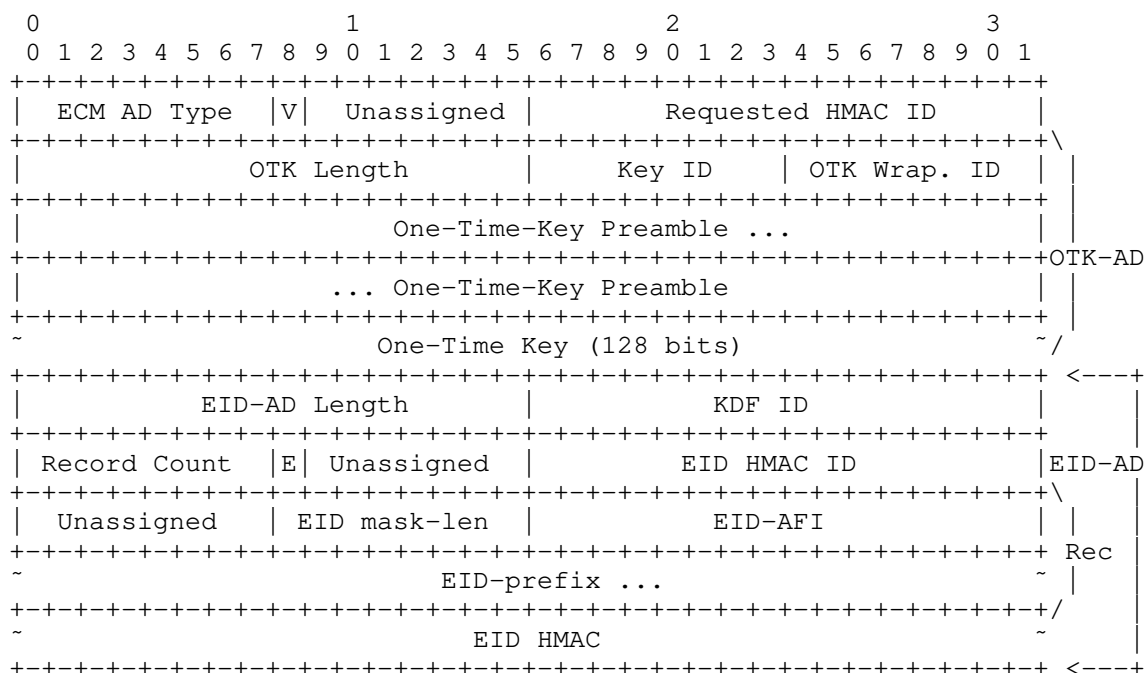


Figure 1: LISP-SEC ECM Authentication Data

ECM AD Type: 1 (LISP-SEC Authentication Data). See Section 8.

V: Key Version bit. This bit is toggled when the sender switches to a new OTK wrapping key

Unassigned: Set to 0 on transmission and ignored on receipt.

Requested HMAC ID: The HMAC algorithm, that will be used to protect the mappings, requested by the ITR. See Section 6.4 for details, and Section 8.3 for HMAC IDs that MUST be supported.

OTK Length: The length (in bytes) of the OTK Authentication Data (OTK-AD), that contains the OTK Preamble and the OTK.

Key ID: The identifier of the pre-shared secret shared by an ITR and the Map-Resolver, and by the Map-Server and an ETR. Per-message keys are derived from the pre-shared secret to encrypt, authenticate the origin and protect the integrity of the OTK. The Key ID allows to rotate between multiple pre-shared secrets in a non disruptive way.

OTK Wrapping ID: The identifier of the key derivation function and of the key wrapping algorithm used to encrypt the One-Time-Key. See Section 6.5 for more details, and Section 8.4 for Wrapping IDs that MUST be supported.

One-Time-Key Preamble: set to 0 if the OTK is not encrypted. When the OTK is encrypted, this field MAY carry additional metadata resulting from the key wrapping operation. When a 128-bit OTK is sent unencrypted by Map-Resolver, the OTK Preamble is set to 0x0000000000000000 (64 bits). See Section 6.5.1 for details.

One-Time-Key: the OTK wrapped as specified by OTK Wrapping ID. See Section 6.5 for details.

EID-AD Length: length (in bytes) of the EID Authentication Data (EID-AD). The ITR MUST set EID-AD Length to 4 bytes, as it only fills the KDF ID field, and all the remaining fields part of the EID-AD are not present. An EID-AD MAY contain multiple EID-records. Each EID-record is 4-byte long plus the length of the AFI-encoded EID-prefix.

KDF ID: Identifier of the Key Derivation Function used to derive the MS-OTK. The ITR MAY use this field to indicate the recommended KDF algorithm, according to local policy. The Map-Server can overwrite the KDF ID if it does not support the KDF ID recommended by the ITR. See Section 5.4 for more details, and Section 8.5 for KDF IDs that MUST be supported.

Record Count: The number of records in this Map-Request message. A record is comprised of the portion of the packet that is labeled 'Rec' above and occurs the number of times equal to Record Count.

E: ETR-Cant-Sign bit. This bit is set to 1 to signal to the ITR that at least one of the ETRs authoritative for the EID prefixes of this Map-Reply has not enabled LISP-SEC. This allows the ITR to securely downgrade to non LISP-SEC requests, as specified in Section 6.7, if so desired.

Unassigned: Set to 0 on transmission and ignored on receipt.

EID HMAC ID: Identifier of the HMAC algorithm used to protect the integrity of the EID-AD. This field is filled by Map-Server that computed the EID-prefix HMAC. See Section 5.4 for more details, and Section 8.3 for HMAC IDs that MUST be supported.

EID mask-len: Mask length for EID-prefix.

EID-AFI: Address family of EID-prefix according to [AFN].

EID-prefix: The Map-Server uses this field to specify the EID-prefix that the destination ETR is authoritative for, and is the longest match for the requested EID.

EID HMAC: HMAC of the EID-AD computed and inserted by Map-Server. Before computing the HMAC operation the EID HMAC field MUST be set to 0. The HMAC MUST cover the entire EID-AD.

6.2. Map-Reply LISP-SEC Extensions

LISP-SEC uses the Map-Reply defined in [I-D.ietf-lisp-rfc6833bis], with Type set to 2, and S-bit set to 1 to indicate that the Map-Reply message includes Authentication Data (AD). The format of the LISP-SEC Map-Reply Authentication Data is defined in Figure 2. PKT-AD is the Packet Authentication Data that covers the Map-Reply payload.

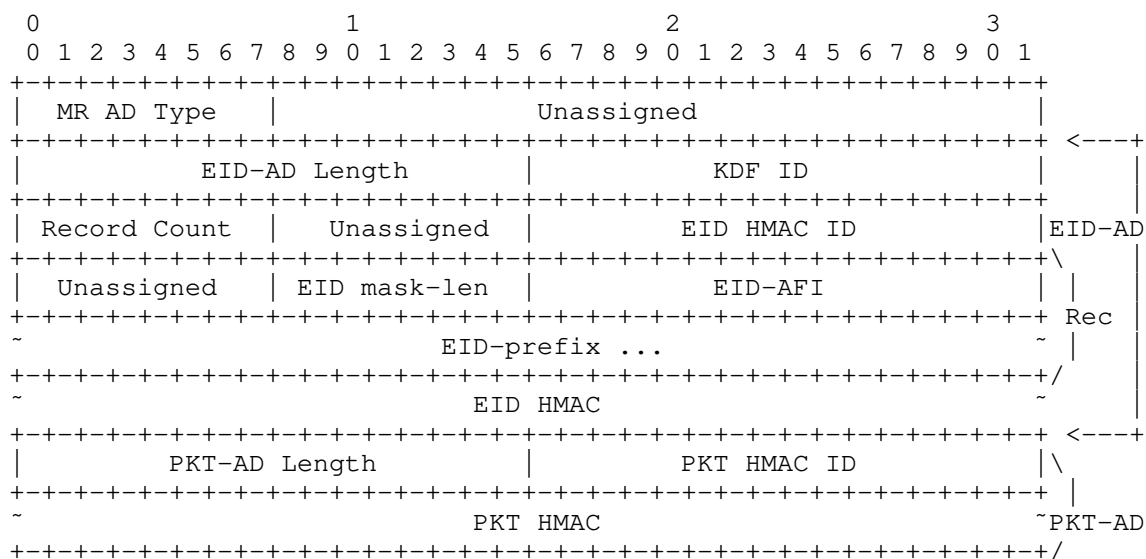


Figure 2: LISP-SEC Map-Reply Authentication Data

MR AD Type: 1 (LISP-SEC Authentication Data). See Section 8.

EID-AD Length: length (in bytes) of the EID-AD. An EID-AD MAY contain multiple EID-records. Each EID-record is 4-byte long plus the length of the AFI-encoded EID-prefix.

KDF ID: Identifier of the Key Derivation Function used to derive MS-OTK. See Section 6.7 for more details, and Section 8.5 for KDF IDs that MUST be supported.

Record Count: The number of records in this Map-Reply message. A record is comprised of the portion of the packet that is labeled 'Rec' above and occurs the number of times equal to Record Count.

Unassigned: Set to 0 on transmission and ignored on receipt.

EID HMAC ID: Identifier of the HMAC algorithm used to protect the integrity of the EID-AD. See Section 6.7 for more details, and Section 8.3 for HMAC IDs that MUST be supported.

EID mask-len: Mask length for EID-prefix.

EID-AFI: Address family of EID-prefix according to [RFC8060].

EID-prefix: This field contains an EID-prefix that the destination ETR is authoritative for, and is the longest match for the requested EID.

EID HMAC: HMAC of the EID-AD, as computed by the Map-Server. Before computing the HMAC operation the EID HMAC field MUST be set to 0. The HMAC covers the entire EID-AD.

PKT-AD Length: length (in bytes) of the Packet Authentication Data (PKT-AD).

PKT HMAC ID: Identifier of the HMAC algorithm used to protect the integrity of the Map-Reply. See Section 8.3 for HMAC IDs that MUST be supported.

PKT HMAC: HMAC of the whole Map-Reply packet, including the LISP-SEC Authentication Data. The scope of the authentication goes from the Map-Reply Type field to the PKT HMAC field included. Before computing the HMAC operation the PKT HMAC field MUST be set to 0. See Section 6.8 for more details.

6.3. Map-Register LISP-SEC Extensions

This memo is allocating one of the bits marked as Unassigned in the Map-Register message defined in [I-D.ietf-lisp-rfc6833bis]. More precisely, the second bit after the Type field in a Map-Register message is allocated as the S bit. The S bit indicates to the Map-Server that the registering ETR is LISP-SEC enabled. An ETR that supports LISP-SEC MUST set the S bit in its Map-Register messages.

6.4. ITR Processing: Generating a Map-Request

Upon creating a Map-Request, the ITR generates a random ITR-OTK that is stored locally (until the corresponding Map-Reply is received), together with the nonce generated as specified in [I-D.ietf-lisp-rfc6833bis].

ITR-OTK confidentiality and integrity protection MUST be provided in the path between the ITR and the Map-Resolver. This can be achieved either by encrypting the ITR-OTK with the pre-shared secret known to the ITR and the Map-Resolver (see Section 6.5), or by enabling DTLS between the ITR and the Map-Resolver.

The Map-Request MUST be encapsulated in an ECM, with the S-bit set to 1, to indicate the presence of Authentication Data.

ITR-OTK is wrapped with the algorithm specified by the OTK Wrapping ID field. See Section 6.5 for further details on OTK encryption. If the NULL-KEY-WRAP-128 algorithm is selected and DTLS is not enabled in the path between the ITR and the Map-Resolver, the Map-Request MUST be dropped and an appropriate log action SHOULD be taken.

The Requested HMAC ID field contains the suggested HMAC algorithm to be used by the Map-Server and the ETR to protect the integrity of the ECM Authentication data and of the Map-Reply. A HMAC ID Value of NONE (0), MAY be used to specify that the ITR has no preferred HMAC ID.

The KDF ID field specifies the suggested key derivation function to be used by the Map-Server to derive the MS-OTK. A KDF Value of NONE (0), MAY be used to specify that the ITR has no preferred KDF ID.

The EID-AD length is set to 4 bytes, since the Authentication Data does not contain EID-prefix Authentication Data, and the EID-AD contains only the KDF ID field.

6.4.1. PITR Processing

The processing performed by a PITR is equivalent to the processing of an ITR. However, if the PITR is directly connected to a Mapping System such as LISP+ALT [RFC6836], the PITR performs the functions of both the ITR and the Map-Resolver forwarding the Map-Request encapsulated in an ECM header that includes the Authentication Data fields as described in Section 6.6.

6.5. Encrypting and Decrypting an OTK

MS-OTK confidentiality and integrity protection MUST be provided in the path between the Map-Server and the ETR. This can be achieved either by enabling DTLS between the Map-Server and the ETR or by encrypting the MS-OTK with the pre-shared secret known to the Map-Server and the ETR [I-D.ietf-lisp-rfc6833bis].

Similarly, ITR-OTK confidentiality and integrity protection MUST be provided in the path between the ITR and the Map-Resolver. This can be achieved either by enabling DTLS between the Map-Server and the ITR, or by encrypting the ITR-OTK with the pre-shared secret known to the ITR and the Map-Resolver. The ITR/Map-Resolver pre-shared key is similar to the Map-Server/ETR pre-shared key.

This section describes OTK processing in the ITR/Map-Resolver path, as well as in the Map-Server/ETR path.

It's important to note that, to prevent ETR's overclaiming attacks, the ITR/Map-Resolver pre-shared secret MUST be different from the Map-Server/ETR pre-shared secret.

The OTK is wrapped using the algorithm specified in the OTK Wrapping ID field. This field identifies both the:

- * Key Encryption Algorithm used to encrypt the wrapped OTK, as well as the
- * Key Derivation Function used to derive a per-message encryption key.

Implementations of this specification MUST support OTK Wrapping ID AES-KEY-WRAP-128+HKDF-SHA256 that specifies the use of the HKDF-SHA256 Key Derivation Function specified in[RFC4868] to derive a per-message encryption key (per-msg-key), as well as the AES-KEY-WRAP-128 Key Wrap algorithm used to encrypt a 128-bit OTK, according to [RFC3394].

The key wrapping process for OTK Wrapping ID AES-KEY-WRAP-128+HKDF-SHA256 is described below:

1. The KDF algorithm is identified by the field 'OTK Wrapping ID' according to the table in Section 8.4.
2. The Key Wrap algorithm is identified by the field 'OTK Wrapping ID' according to the table in Section 8.4.

3. If the NULL-KEY-WRAP-128 algorithm (defined in (Section 8.4)) is selected and DTLS is not enabled, the Map-Request MUST be dropped and an appropriate log action SHOULD be taken.
4. The pre-shared secret used to derive the per-msg-key is represented by PSK[Key ID], that is the pre-shared secret identified by the 'Key ID'.
5. The per-message encryption key is computed as:
 - * $\text{per-msg-key} = \text{KDF}(\text{nonce} + s + \text{PSK}[\text{Key ID}])$
 - * where the nonce is the value in the Nonce field of the Map-Request, and
 - * 's' is the string "OTK-Key-Wrap"
6. According to [RFC3394] the per-msg-key is used to wrap the OTK with AES-KEY-WRAP-128. The AES Key Wrap Initialization Value MUST be set to 0xA6A6A6A6A6A6A6A6 (64 bits). The output of the AES Key Wrap operation is 192-bit long. The most significant 64-bit are copied in the One-Time Key Preamble field, while the 128 less significant bits are copied in the One-Time Key field of the LISP-SEC Authentication Data.

When decrypting an encrypted OTK the receiver MUST verify that the Initialization Value resulting from the AES Key Wrap decryption operation is equal to 0xA6A6A6A6A6A6A6A6. If this verification fails the receiver MUST discard the entire message.

6.5.1. Unencrypted OTK

MS-OTK confidentiality and integrity protection MUST be provided in the path between the Map-Server and the ETR. Similarly, ITR-OTK confidentiality and integrity protection MUST be provided in the path between the ITR and the Map-Resolver.

However, when DTLS is enabled the OTK MAY be sent unencrypted as transport layer security is providing confidentiality and integrity protection.

When a 128-bit OTK is sent unencrypted the OTK Wrapping ID is set to NULL_KEY_WRAP_128, and the OTK Preamble is set to 0x0000000000000000 (64 bits).

6.6. Map-Resolver Processing

Upon receiving an encapsulated Map-Request with the S-bit set, the Map-Resolver decapsulates the ECM message. The ITR-OTK, if encrypted, is decrypted as specified in Section 6.5.

Protecting the confidentiality of the ITR-OTK and, in general, the security of how the Map-Request is handed by the Map-Resolver to the Map-Server, is specific to the particular Mapping System used, and outside of the scope of this memo.

In Mapping Systems where the Map-Server is compliant with [I-D.ietf-lisp-rfc6833bis], the Map-Resolver originates a new ECM header with the S-bit set, that contains the unencrypted ITR-OTK, as specified in Section 6.5, and the other data derived from the ECM Authentication Data of the received encapsulated Map-Request.

The Map-Resolver then forwards to the Map-Server the received Map-Request, encapsulated in the new ECM header that includes the newly computed Authentication Data fields.

6.7. Map-Server Processing

Upon receiving an ECM encapsulated Map-Request with the S-bit set to 1, the Map-Server process the Map-Request according to the value of the security-capable S-bit and of the proxy map-reply P-bit contained in the Map-Register sent by the ETRs authoritative for that prefix during registration.

Processing of the Map-Request MUST proceed in the order described in the table below, applying the processing corresponding to the first rule that matches the conditions indicated in the first column:

Matching Condition	Processing
1. At least one of the ETRs authoritative for the EID prefix included in the Map-Request registered with the P-bit set to 1	The Map-Server MUST generate a LISP-SEC protected Map-Reply as specified in Section 6.7.2. The ETR-Cant-Sign E-bit in the EID Authentication Data (EID-AD) MUST be set to 0.
2. At least one of the ETRs authoritative for the EID prefix included in the Map-Request registered with the S-bit set to 1	The Map-Server MUST generate a LISP-SEC protected Encapsulated Map-Request (as specified in Section 6.7.1), to be sent to one of the authoritative ETRs that registered with the S-bit set to 1 (and the P-bit set to 0). If there is at least one ETR that registered with the S-bit set to 0, the ETR-Cant-Sign E-bit of the EID-AD MUST be set to 1 to signal the ITR that a non LISP-SEC Map-Request might reach additional ETRs that have LISP-SEC disabled.
3. All the ETRs authoritative for the EID prefix included in the Map-Request registered with the S-bit set to 0	The Map-Server MUST send a Negative Map-Reply protected with LISP-SEC, as described in Section 6.7.2. The ETR-Cant-Sign E-bit MUST be set to 1 to signal the ITR that a non LISP-SEC Map-Request might reach additional ETRs that have LISP-SEC disabled.

Table 1

In this way the ITR that sent a LISP-SEC protected Map-Request always receives a LISP-SEC protected Map-Reply. However, the ETR-Cant-Sign E-bit set to 1 specifies that a non LISP-SEC Map-Request might reach additional ETRs that have LISP-SEC disabled. This mechanism allows the ITR to securely downgrade to non LISP-SEC requests, if so desired.

6.7.1. Generating a LISP-SEC Protected Encapsulated Map-Request

The Map-Server decapsulates the ECM and generates a new ECM Authentication Data. The Authentication Data includes the OTK-AD and the EID-AD, that contains EID-prefix authorization information, that are eventually received by the requesting ITR.

The Map-Server updates the OTK-AD by deriving a new OTK (MS-OTK) from the ITR-OTK received with the Map-Request. MS-OTK is derived applying the key derivation function specified in the KDF ID field. If the algorithm specified in the KDF ID field is not supported, the Map-Server uses a different algorithm to derive the key and updates the KDF ID field accordingly.

MS-OTK confidentiality and integrity protection MUST be provided in the path between the Map-Server and the ETR. This can be achieved either by enabling DTLS between the Map-Server and the ETR, or by encrypting the MS-OTK with the pre-shared secret known to the Map-Server and the ETR.

The Map-Request MUST be encapsulated in an ECM, with the S-bit set to 1, to indicate the presence of Authentication Data.

MS-OTK is wrapped with the algorithm specified by the OTK Wrapping ID field. See Section 6.5 for further details on OTK encryption. If the NULL-KEY-WRAP-128 algorithm is selected and DTLS is not enabled in the path between the Map-Server and the ETR, the Map-Request MUST be dropped and an appropriate log action SHOULD be taken.

The Map-Server includes in the EID-AD the longest match registered EID-prefix for the destination EID, and an HMAC of this EID-prefix. The HMAC is keyed with the ITR-OTK contained in the received ECM Authentication Data, and the HMAC algorithm is chosen according to the Requested HMAC ID field. If the Map-Server does not support this algorithm, the Map-Server uses a different algorithm and specifies it in the EID HMAC ID field. The scope of the HMAC operation covers the entire EID-AD, from the EID-AD Length field to the EID HMAC field, which must be set to 0 before the computation.

The Map-Server then forwards the updated ECM encapsulated Map-Request, that contains the OTK-AD, the EID-AD, and the received Map-Request to an authoritative ETR as specified in [I-D.ietf-lisp-rfc6833bis].

6.7.2. Generating a Proxy Map-Reply

LISP-SEC proxy Map-Reply are generated according to [I-D.ietf-lisp-rfc6833bis], with the Map-Reply S-bit set to 1. The Map-Reply includes the Authentication Data that contains the EID-AD, computed as specified in Section 6.7.1, as well as the PKT-AD computed as specified in Section 6.8.

6.8. ETR Processing

Upon receiving an ECM encapsulated Map-Request with the S-bit set, the ETR decapsulates the ECM message. The OTK field, if encrypted, is decrypted as specified in Section 6.5 to obtain the unencrypted MS-OTK.

The ETR then generates a Map-Reply as specified in [I-D.ietf-lisp-rfc6833bis] and includes the Authentication Data that contains the EID-AD, as received in the encapsulated Map-Request, as well as the PKT-AD.

The EID-AD is copied from the Authentication Data of the received encapsulated Map-Request.

The PKT-AD contains the HMAC of the whole Map-Reply packet, keyed with the MS-OTK and computed using the HMAC algorithm specified in the Requested HMAC ID field of the received encapsulated Map-Request. If the ETR does not support the Requested HMAC ID, it uses a different algorithm and updates the PKT HMAC ID field accordingly. The scope of the HMAC operation covers the entire PKT-AD, from the Map-Reply Type field to the PKT HMAC field, which must be set to 0 before the computation.

Finally the ETR sends the Map-Reply to the requesting ITR as specified in [I-D.ietf-lisp-rfc6833bis].

6.9. ITR Processing: Receiving a Map-Reply

In response to an encapsulated Map-Request that has the S-bit set, an ITR MUST receive a Map-Reply with the S-bit set, that includes an EID-AD and a PKT-AD. If the Map-Reply does not include both ADs, the ITR MUST discard it. In response to an encapsulated Map-Request with S-bit set to 0, the ITR expects a Map-Reply with S-bit set to 0, and the ITR SHOULD discard the Map-Reply if the S-bit is set.

Upon receiving a Map-Reply, the ITR must verify the integrity of both the EID-AD and the PKT-AD, and MUST discard the Map-Reply if one of the integrity checks fails. After processing the Map-Reply, the ITR must discard the <nonce, ITR-OTK> pair associated to the Map-Reply

The integrity of the EID-AD is verified using the ITR-OTK (stored locally for the duration of this exchange) to re-compute the HMAC of the EID-AD using the algorithm specified in the EID HMAC ID field. If the EID HMAC ID field does not match the Requested HMAC ID the ITR SHOULD discard the Map-Reply and send, at the first opportunity it needs to, a new Map-Request with a different Requested HMAC ID field, according to ITR's local policy. The scope of the HMAC operation covers the entire EID-AD, from the EID-AD Length field to the EID HMAC field.

ITR MUST set the EID HMAC ID field to 0 before computing the HMAC.

To verify the integrity of the PKT-AD, first the MS-OTK is derived from the locally stored ITR-OTK using the algorithm specified in the KDF ID field. This is because the PKT-AD is generated by the ETR using the MS-OTK. If the KDF ID in the Map-Reply does not match the KDF ID requested in the Map-Request, the ITR SHOULD discard the Map-Reply and send, at the first opportunity it needs to, a new Map-Request with a different KDF ID, according to ITR's local policy. Without consistent configuration of involved entities, extra delays may be experienced. However, since HKDF-SHA1-128 is specified as mandatory to implement in Section 8.5, the process will eventually converge.

The derived MS-OTK is then used to re-compute the HMAC of the PKT-AD using the Algorithm specified in the PKT HMAC ID field. If the PKT HMAC ID field does not match the Requested HMAC ID the ITR SHOULD discard the Map-Reply and send, at the first opportunity it needs to, a new Map-Request with a different Requested HMAC ID according to ITR's local policy or until all HMAC IDs supported by the ITR have been attempted.

Each individual Map-Reply EID-record is considered valid only if: (1) both EID-AD and PKT-AD are valid, and (2) the intersection of the EID-prefix in the Map-Reply EID-record with one of the EID-prefixes contained in the EID-AD is not empty. After identifying the Map-Reply record as valid, the ITR sets the EID-prefix in the Map-Reply record to the value of the intersection set computed before, and adds the Map-Reply EID-record to its EID-to-RLLOC cache, as described in [I-D.ietf-lisp-rfc6833bis]. An example of Map-Reply record validation is provided in Section 6.9.1.

[I-D.ietf-lisp-rfc6833bis] allows ITRs to send solicited Map-Requests directly to the ETRs that send the SMR message in the first place. However, in order to receive a secure Map-Reply, ITRs SHOULD send SMR triggered Map-Requests over the mapping system using the specifications of this memo. If an ITR accepts piggybacked Map-Replies, it SHOULD also send a Map-Request over the mapping system in order to verify the piggybacked Map-Reply with a secure Map-Reply.

6.9.1. Map-Reply Record Validation

The payload of a Map-Reply may contain multiple EID-records. The whole Map-Reply is signed by the ETR, with the PKT HMAC, to provide integrity protection and origin authentication to the EID-prefix records claimed by the ETR. The Authentication Data field of a Map-Reply may contain multiple EID-records in the EID-AD. The EID-AD is signed by the Map-Server, with the EID HMAC, to provide integrity protection and origin authentication to the EID-prefix records inserted by the Map-Server.

Upon receiving a Map-Reply with the S-bit set, the ITR first checks the validity of both the EID HMAC and of the PKT-AD HMAC. If either one of the HMACs is not valid, a log action MUST be taken and the Map-Reply MUST NOT be processed any further. If both HMACs are valid, the ITR proceeds with validating each individual EID-record claimed by the ETR by computing the intersection of each one of the EID-prefix contained in the payload of the Map-Reply with each one of the EID-prefixes contained in the EID-AD. An EID-record is valid only if at least one of the intersections is not the empty set.

For instance, the Map-Reply payload contains 3 mapping record EID-prefixes:

2001:db8:102::/48

2001:db8:103::/48

2001:db8:200::/40

The EID-AD contains two EID-prefixes:

2001:db8:103::/48

2001:db8:203::/48

The EID-record with EID-prefix 2001:db8:102::/48 is not eligible to be used by the ITR since it is not included in any of the EID-ADs signed by the Map-Server. A log action MUST be taken.

The EID-record with EID-prefix 2001:db8:103::/48 is eligible to be used by the ITR because it matches the second EID-prefix contained in the EID-AD.

The EID-record with EID-prefix 2001:db8:200::/40 is not eligible to be used by the ITR since it is not included in any of the EID-ADs signed by the Map-Server. A log action **MUST** be taken. In this last example the ETR is trying to over claim the EID-prefix 2001:db8:200::/40, but the Map-Server authorized only 2001:db8:203::/48, hence the EID-record is discarded.

7. Security Considerations

7.1. Mapping System Security

The LISP-SEC threat model described in Section 4, assumes that the LISP Mapping System is working properly and eventually delivers Map-Request messages to a Map-Server that is authoritative for the requested EID.

It is assumed that the Mapping System ensures the confidentiality of the OTK, and the integrity of the Map-Reply data. However, how the LISP Mapping System is secured is out of the scope of this document.

Similarly, Map-Register security, including the right for a LISP entity to register an EID-prefix or to claim presence at an RLOC, is out of the scope of LISP-SEC.

7.2. Random Number Generation

The ITR-OTK **MUST** be generated by a properly seeded pseudo-random (or strong random) source. See [RFC4086] for advice on generating security-sensitive random data

7.3. Map-Server and ETR Colocation

If the Map-Server and the ETR are colocated, LISP-SEC does not provide protection from overclaiming attacks mounted by the ETR. However, in this particular case, since the ETR is within the trust boundaries of the Map-Server, ETR's overclaiming attacks are not included in the threat model.

7.4. Deploying LISP-SEC

This memo is written according to [RFC2119]. Specifically, the use of the key word SHOULD "or the adjective 'RECOMMENDED', mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course".

Those deploying LISP-SEC according to this memo, should carefully weight how the LISP-SEC threat model applies to their particular use case or deployment. If they decide to ignore a particular recommendation, they should make sure the risk associated with the corresponding threats is well understood.

As an example, in certain closed and controlled deployments, it is possible that the threat associated with a MITM between the xTR and the Mapping System is very low, and after carfeul consideration it may be decided to allow a NULL key wrapping algorithm while carrying the OTKs between the xTR and the Mapping System.

As an example at the other end of the spectrum, in certain other deployments, attackers may be very sophisticated, and force the deployers to enforce very strict policies in term of HMAC algorithms accepted by an ITR.

Similar considerations apply to the entire LISP-SEC threat model, and should guide the deployers and implementors whenever they encounter the key word SHOULD across this memo.

7.5. Shared Keys Provisioning

Provisioning of the keys shared between the ITR and the Map-Resolver as well as between the ETR and the Map-Server should be performed via an orchestration infrastructure and it is out of the scope of this memo. It is recommended that both shared keys are refreshed at periodical intervals to address key aging or attackers gaining unauthorized access to the shared keys. Shared keys should be unpredictable random values.

7.6. Replay Attacks

An attacker can capture a valid Map-Request and/or Map-Reply and replay it, however once the ITR receives the original Map-Reply the <nonce,ITR-OTK> pair stored at the ITR will be discarded. If a replayed Map-Reply arrives at the ITR, there is no <nonce,ITR-OTK> that matches the incoming Map-Reply and will be discarded.

In case of replayed Map-Request, the Map-Server, Map-Resolver and ETR will have to do a LISP-SEC computation. This is equivalent to a valid LISP-SEC computation and an attacker does not obtain any benefit.

7.7. Message Privacy

DTLS [RFC6347] SHOULD be used to provide communication privacy and to prevent eavesdropping, tampering, or message forgery to the messages exchanged between the ITR, Map-Resolver, Map-Server, and ETR.

7.8. Denial of Service and Distributed Denial of Service Attacks

LISP-SEC mitigates the risks of Denial of Service and Distributed Denial of Service attacks by protecting the integrity and authenticating the origin of the Map-Request/Map-Reply messages, and by preventing malicious ETRs from overclaiming EID prefixes that could re-direct traffic directed to a potentially large number of hosts.

8. IANA Considerations

8.1. ECM AD Type Registry

IANA is requested to create the "ECM Authentication Data Type" registry with values 0-255, for use in the ECM LISP-SEC Extensions Section 6.1. The registry MUST be initially populated with the following values:

Name	Value	Defined In
Reserved	0	This memo
LISP-SEC-ECM-EXT	1	This memo

Values 2-255 are unassigned. They are to be assigned according to the "Specification Required" policy defined in [RFC8126].

8.2. Map-Reply AD Type Registry

IANA is requested to create the "Map-Reply Authentication Data Type" registry with values 0-255, for use in the Map-Reply LISP-SEC Extensions Section 6.2. The registry MUST be initially populated with the following values:

Name	Value	Defined In
Reserved	0	This memo
LISP-SEC-MR-EXT	1	This memo

Values 2-255 are unassigned. They are to be assigned according to the "Specification Required" policy defined in [RFC8126].

8.3. HMAC Functions

IANA is requested to create the "LISP-SEC Authentication Data HMAC ID" registry with values 0-65535 for use as Requested HMAC ID, EID HMAC ID, and PKT HMAC ID in the LISP-SEC Authentication Data:

Name	Number	Defined In
NONE	0	This memo
AUTH-HMAC-SHA-1-96	1	[RFC2104]
AUTH-HMAC-SHA-256-128	2	[RFC6234]

Values 3-65535 are unassigned. They are to be assigned according to the "Specification Required" policy defined in [RFC8126].

AUTH-HMAC-SHA-1-96 MUST be supported, AUTH-HMAC-SHA-256-128 SHOULD be supported.

8.4. Key Wrap Functions

IANA is requested to create the "LISP-SEC Authentication Data Key Wrap ID" registry with values 0-65535 for use as OTK key wrap algorithms ID in the LISP-SEC Authentication Data:

Name	Number	KEY WRAP	KDF
Reserved	0	None	None
NULL-KEY-WRAP-128	1	This memo	None
AES-KEY-WRAP-128+HKDF-SHA256	2	[RFC3394]	[RFC4868]

Values 3-65535 are unassigned. They are to be assigned according to the "Specification Required" policy defined in [RFC8126].

NULL-KEY-WRAP-128, and AES-KEY-WRAP-128+HKDF-SHA256 MUST be supported.

NULL-KEY-WRAP-128 is used to carry an unencrypted 128-bit OTK, with a 64-bit preamble set to 0x0000000000000000 (64 bits).

8.5. Key Derivation Functions

IANA is requested to create the "LISP-SEC Authentication Data Key Derivation Function ID" registry with values 0-65535 for use as KDF ID in the LISP-SEC Authentication Data:

Name	Number	Defined In
NONE	0	This memo
HKDF-SHA1-128	1	[RFC5869]

Values 2-65535 are unassigned. They are to be assigned according to the "Specification Required" policy defined in [RFC8126].

HKDF-SHA1-128 MUST be supported

9. Acknowledgements

The authors would like to acknowledge Pere Monclus, Dave Meyer, Dino Farinacci, Brian Weis, David McGrew, Darrel Lewis and Landon Curt Noll for their valuable suggestions provided during the preparation of this document.

10. References

10.1. Normative References

- [I-D.ietf-lisp-rfc6833bis]
Farinacci, D., Maino, F., Fuller, V., and A. Cabellos,
"Locator/ID Separation Protocol (LISP) Control-Plane",
Work in Progress, Internet-Draft, draft-ietf-lisp-
rfc6833bis-30, 18 November 2020,
<[https://www.ietf.org/archive/id/draft-ietf-lisp-
rfc6833bis-30.txt](https://www.ietf.org/archive/id/draft-ietf-lisp-rfc6833bis-30.txt)>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker,
"Randomness Requirements for Security", BCP 106, RFC 4086,
DOI 10.17487/RFC4086, June 2005,
<<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-
384, and HMAC-SHA-512 with IPsec", RFC 4868,
DOI 10.17487/RFC4868, May 2007,
<<https://www.rfc-editor.org/info/rfc4868>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.

- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

10.2. Informational References

- [AFN] IANA - Internet Assigned Numbers Authority, "Address Family Numbers", 2021, <<http://www.iana.org/assignments/address-family-numbers/>>.
- [I-D.ietf-lisp-rfc6830bis] Farinacci, D., Fuller, V., Meyer, D., Lewis, D., and A. Cabellos, "The Locator/ID Separation Protocol (LISP)", Work in Progress, Internet-Draft, draft-ietf-lisp-rfc6830bis-36, 18 November 2020, <<https://www.ietf.org/archive/id/draft-ietf-lisp-rfc6830bis-36.txt>>.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <<https://www.rfc-editor.org/info/rfc2104>>.
- [RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394, September 2002, <<https://www.rfc-editor.org/info/rfc3394>>.
- [RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <<https://www.rfc-editor.org/info/rfc5869>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, DOI 10.17487/RFC6836, January 2013, <<https://www.rfc-editor.org/info/rfc6836>>.

[RFC7835] Saucez, D., Iannone, L., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Threat Analysis", RFC 7835, DOI 10.17487/RFC7835, April 2016, <<https://www.rfc-editor.org/info/rfc7835>>.

[RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.

Authors' Addresses

Fabio Maino
Cisco Systems
170 Tasman Drive
San Jose, California 95134
United States of America

Email: fmaino@cisco.com

Vina Ermagan
Google
California
United States of America

Email: ermagan@gmail.com

Albert Cabellos
Universitat Politecnica de Catalunya
c/ Jordi Girona s/n
08034 Barcelona
Spain

Email: acabello@ac.upc.edu

Damien Saucez
Inria
2004 route des Lucioles - BP 93
Sophia Antipolis
France

Email: damien.saucez@inria.fr

LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: September 7, 2019

V. Ermagan
Google
A. Rodriguez-Natal
F. Coras
C. Moberg
R. Rahman
Cisco Systems
A. Cabellos-Aparicio
Technical University of Catalonia
F. Maino
Cisco Systems
March 6, 2019

LISP YANG Model
draft-ietf-lisp-yang-11

Abstract

This document describes a YANG data model to use with the Locator/ID Separation Protocol (LISP).

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 7, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language	3
1.2. Tree Diagrams	3
2. LISP Module	3
2.1. Module Structure	3
2.2. Module Definition	6
3. LISP-ITR Module	16
3.1. Module Structure	17
3.2. Module Definition	22
4. LISP-ETR Module	26
4.1. Module Structure	26
4.2. Module Definition	28
5. LISP-Map-Server Module	32
5.1. Module Structure	33
5.2. Module Definition	41
6. LISP-Map-Resolver Module	47
6.1. Module Structure	47
6.2. Module Definition	48
7. LISP-Address-Types Module	50
7.1. Module Definition	50
7.2. Data Model examples	64
7.2.1. LISP protocol instance	64
7.2.2. LISP ITR	66
7.2.3. LISP ETR	66
7.2.4. LISP Map-Server	69
8. Acknowledgments	70
9. IANA Considerations	70
10. Security Considerations	72
11. Normative References	75
Authors' Addresses	76

1. Introduction

The Locator/ID Separation Protocol (LISP) defines several network elements subject to be configured. This document presents the YANG data models required for basic configuration of all major LISP

[RFC6830] elements. The models also capture some essential operational data elements as well.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Tree Diagrams

This document uses the graphical representation of data models defined in [RFC8340].

2. LISP Module

This module is the base LISP module that is augmented in multiple models to represent various LISP device roles.

2.1. Module Structure

```

module: ietf-lisp
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol:
      +--rw lisp
        +--rw locator-sets
          +--rw locator-set* [locator-set-name]
            +--rw locator-set-name      string
            +--rw (locator-type)?
              +--:(local-interface)
                +--rw interface* [interface-ref]
                  +--rw interface-ref    if:interface-ref
                  +--rw priority?         uint8
                  +--rw weight?           uint8
                  +--rw multicast-priority? uint8
                  +--rw multicast-weight? uint8
              +--:(general-locator)
                +--rw locator* [id]
                  +--rw id                string
                  +--rw locator-address
                    +--rw address-type
                    |   lisp-address-family-ref
                    +--rw (address)?
                      +--:(no-address)
                      |   +--rw no-address?          empty
                      +--:(ipv4)

```



```

|   +--rw ipv4?
|       inet:ipv4-address
+--:(ipv4-prefix)
|   +--rw ipv4-prefix?
|       inet:ipv4-prefix
+--:(ipv6)
|   +--rw ipv6?
|       inet:ipv6-address
+--:(ipv6-prefix)
|   +--rw ipv6-prefix?
|       inet:ipv6-prefix
+--:(mac)
|   +--rw mac?
|       yang:mac-address
+--:(distinguished-name)
|   +--rw distinguished-name?
|       distinguished-name-type
+--:(as-number)
|   +--rw as-number?
|       inet:as-number
+--:(null-address)
|   +--rw null-address
|       +--rw address?    empty
+--:(afi-list)
|   +--rw afi-list
|       +--rw address-list*
|           simple-address
+--:(instance-id)
|   +--rw instance-id
|       +--rw instance-id?
|           | instance-id-type
|       +--rw mask-length?    uint8
|       +--rw address?        simple-address
+--:(as-number-lcaf)
|   +--rw as-number-lcaf
|       +--rw as?            inet:as-number
|       +--rw address?        simple-address
+--:(application-data)
|   +--rw application-data
|       +--rw address?
|           | simple-address
|       +--rw protocol?            uint8
|       +--rw ip-tos?              int32
|       +--rw local-port-low?
|           | inet:port-number
|       +--rw local-port-high?
|           | inet:port-number
|       +--rw remote-port-low?

```

```

|         inet:port-number
|         +--rw remote-port-high?
|         |         inet:port-number
+--:(geo-coordinates)
|   +--rw geo-coordinates
|   |   +--rw latitude?          bits
|   |   +--rw latitude-degrees? uint8
|   |   +--rw latitude-minutes? uint8
|   |   +--rw latitude-seconds? uint8
|   |   +--rw longitude?        bits
|   |   +--rw longitude-degrees? uint16
|   |   +--rw longitude-minutes? uint8
|   |   +--rw longitude-seconds? uint8
|   |   +--rw altitude?         int32
|   |   +--rw address?
|   |       |         simple-address
+--:(nat-traversal)
|   +--rw nat-traversal
|   |   +--rw ms-udp-port?      uint16
|   |   +--rw etr-udp-port?    uint16
|   |   +--rw global-etr-rloc?
|   |       |         simple-address
|   |   +--rw ms-rloc?
|   |       |         simple-address
|   |   +--rw private-etr-rloc?
|   |       |         simple-address
|   |   +--rw rtr-rlocs*
|   |       |         simple-address
+--:(explicit-locator-path)
|   +--rw explicit-locator-path
|   |   +--rw hop* [hop-id]
|   |       |   +--rw hop-id      string
|   |       |   +--rw address?    simple-address
|   |       |   +--rw lrs-bits?   bits
+--:(source-dest-key)
|   +--rw source-dest-key
|   |   +--rw source?    simple-address
|   |   +--rw dest?     simple-address
+--:(key-value-address)
|   +--rw key-value-address
|   |   +--rw key?      simple-address
|   |   +--rw value?    simple-address
+--:(service-path)
|   +--rw service-path
|   |   +--rw service-path-id?
|   |       |         service-path-id-type
|   |   +--rw service-index?    uint8
+--rw priority?                uint8

```

```

|           +--rw weight?                uint8
|           +--rw multicast-priority?    uint8
|           +--rw multicast-weight?      uint8
+--rw lisp-role* [lisp-role-type]
|   +--rw lisp-role-type    lisp-role-ref
+--rw lisp-router-id
|   +--rw site-id?         uint64
|   +--rw xtr-id?          lisp:xtr-id-type
+--rw vpns
|   +--rw vpn* [instance-id]
|       +--rw instance-id    lcaf:instance-id-type
|       +--rw iid-name
|           -> /ni:network-instances/network-instance/name

```

2.2. Module Definition

```

<CODE BEGINS> file "ietf-lisp@2019-03-05.yang"
module ietf-lisp {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp";

  prefix lisp;

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }
  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note
  import ietf-lisp-address-types {
    prefix lcaf;
    reference "RFC XXXX: LISP YANG model";
  }
  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management
      (NMDA version)";
  }
  import ietf-network-instance {
    prefix "ni";
  }

```

```
// RFC Ed.: replace occurrences of YYYY with actual RFC number
// of draft-ietf-rtgwg-ni-model and remove this note
reference
  "RFC YYYY: YANG Model for Network Instances";
}

organization
  "IETF LISP (Locator/ID Separation Protocol) Working Group";
contact
  "WG Web:  <http://tools.ietf.org/wg/lisp/>
  WG List:  <mailto:lisp@ietf.org>

  Editor:    Vina Ermagan
             <mailto:ermagan@gmail.com>

  Editor:    Alberto Rodriguez-Natal
             <mailto:natal@cisco.com>

  Editor:    Reshad Rahman
             <mailto:rrahman@cisco.com>";
description
  "This YANG module defines the generic parameters for LISP.
  The module can be extended by vendors to define vendor-specific
  LISP parameters and policies.

  Copyright (c) 2018 IETF Trust and the persons identified as
  authors of the code.  All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC XXXX; see
  the RFC itself for full legal notices.
  ";

reference "RFC XXXX";

revision 2019-03-05 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6830";
}
```

```
/*
 * Identity definitions
 */
identity lisp {
  base "rt:control-plane-protocol";
  description "LISP protocol.";
  reference
    "RFC 6830: The Locator/ID Separation Protocol (LISP).";
}

identity lisp-role {
  description
    "LISP router role.";
}
identity itr {
  base lisp-role;
  description
    "LISP ITR.";
}
identity pitr {
  base lisp-role;
  description
    "LISP PITR.";
}
identity etr {
  base lisp-role;
  description
    "LISP ETR.";
}
identity petr {
  base lisp-role;
  description
    "LISP PETR.";
}
identity mapping-system {
  description
    "Mapping System interface";
}
identity single-node-mapping-system {
  base mapping-system;
  description
    "logically singular Map Server";
}
typedef mapping-system-ref {
  type identityref {
    base mapping-system;
  }
  description
```

```
    "Mapping System reference";
}

typedef lisp-role-ref {
  type identityref {
    base lisp-role;
  }
  description
    "LISP role reference";
}
typedef map-reply-action {
  type enumeration {
    enum no-action {
      value 0;
      description
        "Mapping is kept alive and no encapsulation occurs.";
    }
    enum natively-forward {
      value 1;
      description
        "Matching packets are not encapsulated or dropped but
        natively forwarded.";
    }
    enum send-map-request {
      value 2;
      description
        "Matching packets invoke Map-Requests.";
    }
    enum drop {
      value 3;
      description
        "Matching packets are dropped.";
    }
  }
  description
    "Defines the lisp map-cache ACT type";
  reference "https://tools.ietf.org/html/rfc6830#section-6.1.4";
}
typedef eid-id {
  type string;
  description
    "Type encoding of lisp-addresses to be generally used in EID
    keyed lists.";
}
typedef auth-algorithm-type {
  type enumeration {
    enum none {
      value 0;
```

```
        description
            "No authentication.";
    }
    enum hmac-sha-1-96 {
        value 1;
        description
            "HMAC-SHA-1-96 (RFC2404) authentication is used.";
    }
    enum hmac-sha-256-128 {
        value 2;
        description
            "HMAC-SHA-256-128 (RFC4868) authentication is used.";
    }
}
description
    "Enumeration of the authentication mechanisms supported by
    LISP.";
reference
    "https://tools.ietf.org/html/rfc6830#section-6.1.6";
}
typedef xtr-id-type {
    type binary {
        length "16";
    }
    description
        "128 bit xTR identifier.";
}

grouping locator-properties {
    description
        "Properties of a RLOC";
    leaf priority {
        type uint8;
        description
            "Locator priority.";
    }
    leaf weight {
        type uint8;
        description
            "Locator weight.";
    }
    leaf multicast-priority {
        type uint8;
        description
            "Locator's multicast priority";
    }
    leaf multicast-weight {
        type uint8;
```

```
        description
          "Locator's multicast weight";
      }
  }

  grouping locators-grouping {
    description
      "Grouping that defines a list of LISP locators.";
    list locator {
      key "id";
      description
        "List of routing locators";
      leaf id {
        type string {
          length "1..64";
        }
        description
          "Locator id";
      }
      container locator-address {
        uses lcaf:lisp-address;
        description
          "The locator address provided in LISP canonincal
            address format.";
      }
      uses locator-properties;
    }
  }

  grouping local-locators-grouping {
    description
      "Grouping that defines a list of LISP locators.";
    list interface {
      key "interface-ref";
      description
        "The address type of the locator";
      leaf interface-ref {
        type if:interface-ref;
        description
          "The name of the interface supporting the locator.";
      }
      uses locator-properties;
    }
  }

  grouping mapping {
    description
```



```
    "Grouping that defines a LISP mapping.";
  container eid {
    uses lcaf:lisp-address;
    description
      "End-host Identifier (EID) to be mapped to a list of
        locators";
  }
  leaf time-to-live {
    type uint32;
    units minutes;
    description
      "Mapping validity period in minutes.";
  }
  leaf creation-time {
    type yang:date-and-time;
    config false;
    description
      "Time when the mapping was created.";
  }
  leaf authoritative {
    type bits {
      bit A {
        description
          "Authoritative bit.";
      }
    }
    description
      "Bit that indicates if mapping comes from an
        authoritative source.";
  }
  leaf static {
    type boolean;
    default "false";
    description
      "This leaf should be true if the mapping is static.";
  }
  choice locator-list {
    description
      "list of locartors are either negative, or positive.";
    case negative-mapping {
      leaf map-reply-action {
        type map-reply-action;
        description
          "Forwarding action for a negative mapping.";
      }
    }
    case positive-mapping {
      container rlocs {

```

```
        uses locators-grouping;
        description
            "List of locators for a positive mapping.";
    }
}
}

grouping mappings {
    description
        "Grouping that defines a list of LISP mappings.";
    list vpn {
        key "instance-id";
        description
            "VPN to which the mappings belong.";
        leaf instance-id {
            type leafref {
                path "/rt:routing/rt:control-plane-protocols"
                    + "/rt:control-plane-protocol/lisp:lisp"
                    + "/lisp:vpns/lisp:vpn"
                    + "/lisp:instance-id";
            }
            description
                "VPN identifier.";
        }
        container mappings {
            description
                "Mappings within the VPN.";
            list mapping {
                key "id";
                description
                    "List of EID to RLOCs mappings.";
                leaf id {
                    type eid-id;
                    description
                        "Id that uniquely identifies a mapping.";
                }
                uses mapping;
            }
        }
    }
}

grouping auth-key {
    description "Grouping that defines authentication keys.";
    container authentication-keys {
        description "Multiple authentication keys can be defined.";
        list authentication-key {
```

```
    key "auth-key-id";
    description
      "Authentication key parameters.";
    leaf auth-key-id {
      type string;
      description
        "Identifier of the authentication key.";
    }
    leaf-list auth-algorithm-id {
      type lisp:auth-algorithm-type;
      description
        "Authentication algorithm used with the key.";
    }
    leaf auth-key-value {
      type string;
      description
        "Clear text authentication key.";
    }
  }
}

augment "/rt:routing/rt:control-plane-protocols"
  + "/rt:control-plane-protocol" {
  when "derived-from-or-self(rt:type, 'lisp:lisp')" {
    description
      "This augmentation is only valid for a control-plane protocol
        instance of LISP.";
  }
  description "LISP protocol ietf-routing module
    control-plane-protocol augmentation.";

  container lisp {
    description
      "Parameters for the LISP subsystem.";

    container locator-sets {
      description
        "Container that defines a named locator set which can be
          referenced elsewhere.";
      list locator-set {
        key "locator-set-name";
        description
          "Multiple locator sets can be defined.";
        leaf locator-set-name {
          type string {
            length "1..64";
          }
        }
      }
    }
  }
}
```

```
        description
            "Locator set name";
    }
    choice locator-type {
        description
            "Locator sets can be based on local interfaces, or
            general locators.";
        case local-interface {
            uses local-locators-grouping;
            description
                "List of locators in this set based on local
                interfaces.";
        }
        case general-locator {
            uses locators-grouping;
            description
                "List of locators in this set based on lisp-address.";
        }
    }
}

list lisp-role {
    key lisp-role-type;
    description
        "List of lisp device roles such as MS, MR, ITR,
        PITR, ETR or PETR.";
    leaf lisp-role-type {
        type lisp-role-ref;
        description
            "The type of LISP device - identity derived from the
            'lisp-device' base identity.";
    }
}

container lisp-router-id {
    when "../lisp-role/lisp-role-type = 'lisp:itr' or
        ../lisp-role/lisp-role-type = 'lisp:pitr' or
        ../lisp-role/lisp-role-type = 'lisp:etr' or
        ../lisp-role/lisp-role-type = 'lisp:petr'" {
        description "Only when ITR, PITR, ETR or PETR.";
    }
    description
        "Site-ID and xTR-ID of the device.";
    leaf site-id {
        type uint64;
        description "Site ID";
    }
}
```

```

    leaf xtr-id {
      type lisp:xtr-id-type;
      description "xTR ID";
    }
  }

  container vpns {
    when "../lisp-role/lisp-role-type = 'lisp:itr' or
      ../lisp-role/lisp-role-type = 'lisp:pitr' or
      ../lisp-role/lisp-role-type = 'lisp:etr' or
      ../lisp-role/lisp-role-type = 'lisp:petr'" {
      description "Only when ITR, PITR, ETR or PETR.";
    }
    description "VPNs";
    list vpn {
      key instance-id;
      unique "iid-name";
      description "List of VPNs";

      leaf instance-id {
        type lcaf:instance-id-type;
        description
          "VPN identifier. The value 0 for instance-id must be used
           for the default VRF.";
      }
      leaf iid-name {
        type leafref {
          path "/ni:network-instances/ni:network-instance/ni:name";
        }
        mandatory true;
        description
          "Name of VPN (e.g. VRF) to which an instance-id is
           bound. Each instance-id is bound to a different VPN";
      }
    }
  }
}
}
<CODE ENDS>

```

3. LISP-ITR Module

This module captures the configuration data model of a LISP ITR. The model also captures some operational data elements.

3.1. Module Structure

```

module: ietf-lisp-itr
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/lisp:lisp:
    +--rw itr!
      +--rw rloc-probing!
        | +--rw interval?          uint16
        | +--rw retries?          uint8
        | +--rw retries-interval? uint16
      +--rw itr-rlocs? leafref
      +--rw map-resolvers
        | +--rw map-resolver* inet:ip-address
      +--rw proxy-etr
        | +--rw proxy-etr-address* inet:ip-address
      +--rw map-cache
        +--ro size?          uint32
        +--ro limit?        uint32
      +--rw vpn* [instance-id]
        +--rw instance-id
          | -> /rt:routing/control-plane-protocols
          |   /control-plane-protocol/lisp:lisp/vpns
          |   /vpn/instance-id
        +--rw mappings
          +--rw mapping* [id]
            +--rw id                      eid-id
            +--rw eid
              +--rw address-type
              |   lisp-address-family-ref
            +--rw (address)?
              +--:(no-address)
              | +--rw no-address?          empty
              +--:(ipv4)
              | +--rw ipv4?
              |   inet:ipv4-address
            +--:(ipv4-prefix)
              | +--rw ipv4-prefix?
              |   inet:ipv4-prefix
            +--:(ipv6)
              | +--rw ipv6?
              |   inet:ipv6-address
            +--:(ipv6-prefix)
              | +--rw ipv6-prefix?
              |   inet:ipv6-prefix
            +--:(mac)
              | +--rw mac?
              |   yang:mac-address
            +--:(distinguished-name)

```

```

|   +--rw distinguished-name?
|       distinguished-name-type
+--:(as-number)
|   +--rw as-number?
|       inet:as-number
+--:(null-address)
|   +--rw null-address
|       +--rw address?    empty
+--:(afi-list)
|   +--rw afi-list
|       +--rw address-list*  simple-address
+--:(instance-id)
|   +--rw instance-id
|       +--rw instance-id?   instance-id-type
|       +--rw mask-length?   uint8
|       +--rw address?       simple-address
+--:(as-number-lcaf)
|   +--rw as-number-lcaf
|       +--rw as?            inet:as-number
|       +--rw address?       simple-address
+--:(application-data)
|   +--rw application-data
|       +--rw address?
|           |   simple-address
+--rw protocol?           uint8
+--rw ip-tos?              int32
+--rw local-port-low?
|   |   inet:port-number
+--rw local-port-high?
|   |   inet:port-number
+--rw remote-port-low?
|   |   inet:port-number
+--rw remote-port-high?
|   |   inet:port-number
+--:(geo-coordinates)
|   +--rw geo-coordinates
|       +--rw latitude?           bits
|       +--rw latitude-degrees?   uint8
|       +--rw latitude-minutes?   uint8
|       +--rw latitude-seconds?   uint8
|       +--rw longitude?          bits
|       +--rw longitude-degrees?  uint16
|       +--rw longitude-minutes?  uint8
|       +--rw longitude-seconds?  uint8
|       +--rw altitude?           int32
|       +--rw address?
|           |   simple-address
+--:(nat-traversal)

```

```

    +--rw nat-traversal
      +--rw ms-udp-port?          uint16
      +--rw etr-udp-port?        uint16
      +--rw global-etr-rloc?
      |   simple-address
      +--rw ms-rloc?
      |   simple-address
      +--rw private-etr-rloc?
      |   simple-address
      +--rw rtr-rlocs*
      |   simple-address
+---:(explicit-locator-path)
  +--rw explicit-locator-path
    +--rw hop* [hop-id]
      +--rw hop-id          string
      +--rw address?        simple-address
      +--rw lrs-bits?       bits
+---:(source-dest-key)
  +--rw source-dest-key
    +--rw source?          simple-address
    +--rw dest?            simple-address
+---:(key-value-address)
  +--rw key-value-address
    +--rw key?              simple-address
    +--rw value?            simple-address
+---:(service-path)
  +--rw service-path
    +--rw service-path-id?
    |   service-path-id-type
    +--rw service-index?    uint8
+--rw time-to-live?          uint32
+--ro creation-time?          yang:date-and-time
+--rw authoritative?         bits
+--rw static?                 boolean
+--rw (locator-list)?
  +---:(negative-mapping)
  |   +--rw map-reply-action?  map-reply-action
  +---:(positive-mapping)
  +--rw rlocs
    +--rw locator* [id]
      +--rw id                  string
      +--rw locator-address
      |   +--rw address-type
      |   |   lisp-address-family-ref
      |   +--rw (address)?
      |   |   +---:(no-address)
      |   |   |   +--rw no-address?
      |   |   |   empty

```



```

+--:(ipv4)
|   +--rw ipv4?
|       inet:ipv4-address
+--:(ipv4-prefix)
|   +--rw ipv4-prefix?
|       inet:ipv4-prefix
+--:(ipv6)
|   +--rw ipv6?
|       inet:ipv6-address
+--:(ipv6-prefix)
|   +--rw ipv6-prefix?
|       inet:ipv6-prefix
+--:(mac)
|   +--rw mac?
|       yang:mac-address
+--:(distinguished-name)
|   +--rw distinguished-name?
|       distinguished-name-type
+--:(as-number)
|   +--rw as-number?
|       inet:as-number
+--:(null-address)
|   +--rw null-address
|       +--rw address?    empty
+--:(afi-list)
|   +--rw afi-list
|       +--rw address-list*
|           simple-address
+--:(instance-id)
|   +--rw instance-id
|       +--rw instance-id?
|           |   instance-id-type
|       +--rw mask-length?    uint8
|       +--rw address?
|           simple-address
+--:(as-number-lcaf)
|   +--rw as-number-lcaf
|       +--rw as?
|           |   inet:as-number
|       +--rw address?
|           simple-address
+--:(application-data)
|   +--rw application-data
|       +--rw address?
|           |   simple-address
|       +--rw protocol?
|           |   uint8
|       +--rw ip-tos?

```

```

|         int32
+--rw local-port-low?
|         inet:port-number
+--rw local-port-high?
|         inet:port-number
+--rw remote-port-low?
|         inet:port-number
+--rw remote-port-high?
|         inet:port-number
+--:(geo-coordinates)
+--rw geo-coordinates
+--rw latitude?
|         bits
+--rw latitude-degrees?
|         uint8
+--rw latitude-minutes?
|         uint8
+--rw latitude-seconds?
|         uint8
+--rw longitude?
|         bits
+--rw longitude-degrees?
|         uint16
+--rw longitude-minutes?
|         uint8
+--rw longitude-seconds?
|         uint8
+--rw altitude?
|         int32
+--rw address?
|         simple-address
+--:(nat-traversal)
+--rw nat-traversal
+--rw ms-udp-port?
|         uint16
+--rw etr-udp-port?
|         uint16
+--rw global-etr-rloc?
|         simple-address
+--rw ms-rloc?
|         simple-address
+--rw private-etr-rloc?
|         simple-address
+--rw rtr-rlocs*
|         simple-address
+--:(explicit-locator-path)
+--rw explicit-locator-path
+--rw hop* [hop-id]

```

```

|--rw hop-id
|      string
|--rw address?
|      simple-address
|--rw lrs-bits?  bits
+--:(source-dest-key)
|  |--rw source-dest-key
|  |  |--rw source?
|  |  |      simple-address
|  |  |--rw dest?
|  |      simple-address
+--:(key-value-address)
|  |--rw key-value-address
|  |  |--rw key?
|  |  |      simple-address
|  |  |--rw value?
|  |      simple-address
+--:(service-path)
|  |--rw service-path
|  |  |--rw service-path-id?
|  |  |      service-path-id-type
|  |  |--rw service-index?
|  |      uint8
+--rw priority?      uint8
+--rw weight?        uint8
+--rw multicast-priority?  uint8
+--rw multicast-weight?    uint8

```

3.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp-itr@2019-02-23.yang"
module ietf-lisp-itr {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-itr";

  prefix lisp-itr;

  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note
  import ietf-lisp {
    prefix lisp;
    reference "RFC XXXX: LISP YANG model";
  }
  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }
}
```

```
import ietf-routing {
  prefix "rt";
  reference
    "RFC 8349: A YANG Data Model for Routing Management
    (NMDA version)";
}

organization
  "IETF LISP (Locator/ID Separation Protocol) Working Group";
contact
  "WG Web:  <http://tools.ietf.org/wg/lisp/>
  WG List:  <mailto:lisp@ietf.org>

  Editor:    Vina Ermagan
             <mailto:ermagan@gmail.com>

  Editor:    Alberto Rodriguez-Natal
             <mailto:natal@cisco.com>

  Editor:    Reshad Rahman
             <mailto:rrahman@cisco.com>";
description
  "This YANG module defines the generic parameters for a LISP
  ITR. The module can be extended by vendors to define
  vendor-specific parameters and policies.

  Copyright (c) 2018 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC XXXX; see
  the RFC itself for full legal notices.
  ";

reference "RFC XXXX";

revision 2019-02-23 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6830";
}
```

```
augment "/rt:routing/rt:control-plane-protocols"
+ "/rt:control-plane-protocol/lisp:lisp" {
  when "lisp:lisp-role/lisp:lisp-role-type = 'lisp:itr' or
        lisp:lisp-role/lisp:lisp-role-type = 'lisp:pitr'" {
    description
      "Augment is valid when LISP role type is ITR or PITR.";
  }
  description
    "This augments LISP devices list with (P)ITR specific
    parameters.";
  container itr {
    presence "LISP (P)ITR operation enabled";
    description
      "ITR parameters";
    container rloc-probing {
      presence "RLOC probing active";
      description
        "RLOC-probing parameters";
      leaf interval {
        type uint16;
        units "seconds";
        description
          "Interval in seconds for resending the probes";
      }
      leaf retries {
        type uint8;
        description
          "Number of retries for sending the probes";
      }
      leaf retries-interval {
        type uint16;
        units "seconds";
        description
          "Interval in seconds between retries when sending probes.
          The action taken if all retries fail to receive is
          impementation specific.";
      }
    }
  }
  leaf itr-rlocs {
    type leafref {
      path "/rt:routing/rt:control-plane-protocols"
        + "/rt:control-plane-protocol/lisp:lisp"
        + "/lisp:locator-sets/lisp:locator-set"
        + "/lisp:locator-set-name";
    }
    description
      "Reference to a locator set that the (P)ITR includes in
      Map-Requests";
  }
}
```

```
    }
    container map-resolvers {
      description
        "Map-Resolvers that the (P) ITR uses.";
      leaf-list map-resolver {
        type inet:ip-address;
        description
          "Each Map-Resolver within the list of Map-Resolvers.";
      }
    }
    container proxy-etr {
      when "../lisp:lisp-role/lisp:lisp-role-type = 'lisp:itr'" {
        description
          "Container exists only when LISP role type is ITR";
      }
      description
        "Proxy ETRs that the ITR uses.";
      leaf-list proxy-etr-address {
        type inet:ip-address;
        description
          "Proxy ETR RLOC address.";
      }
    }
    container map-cache {
      leaf size {
        type uint32;
        config false;
        description
          "Current number of entries in the EID-to-RLOC map-cache";
      }
      leaf limit {
        type uint32;
        config false;
        description
          "Maximum permissible number of entries in the EID-to-RLOC
            map-cache";
      }
      uses lisp:mappings;
      description
        "EID to RLOCs mappings cache.";
    }
  }
}
}
<CODE ENDS>
```

4. LISP-ETR Module

This module captures the configuration data model of a LISP ETR. The model also captures some operational data elements.

4.1. Module Structure

```

module: ietf-lisp-etr
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/lisp:lisp:
      +--rw etr!
        +--rw map-servers
          +--rw map-server* [ms-address]
            +--rw ms-address          inet:ip-address
            +--rw authentication-keys
              +--rw authentication-key* [auth-key-id]
                +--rw auth-key-id      string
                +--rw auth-algorithm-id* lisp:auth-algorithm-type
                +--rw auth-key-value?   string
        +--rw local-eids
          +--rw vpn* [instance-id]
            +--rw instance-id
              -> /rt:routing/control-plane-protocols
                /control-plane-protocol/lisp:lisp/vpns
                /vpn/instance-id
          +--rw eids
            +--rw local-eid* [id]
              +--rw id                  lisp:eid-id
              +--rw eid-address
                +--rw address-type
                |   lisp-address-family-ref
                +--rw (address)?
                  +--:(no-address)
                  |   +--rw no-address?          empty
                  +--:(ipv4)
                  |   +--rw ipv4?
                  |       inet:ipv4-address
                  +--:(ipv4-prefix)
                  |   +--rw ipv4-prefix?
                  |       inet:ipv4-prefix
                  +--:(ipv6)
                  |   +--rw ipv6?
                  |       inet:ipv6-address
                  +--:(ipv6-prefix)
                  |   +--rw ipv6-prefix?
                  |       inet:ipv6-prefix
                  +--:(mac)
                  |   +--rw mac?

```

```

|           yang:mac-address
+---:(distinguished-name)
|   +---rw distinguished-name?
|       distinguished-name-type
+---:(as-number)
|   +---rw as-number?
|       inet:as-number
+---:(null-address)
|   +---rw null-address
|       +---rw address?    empty
+---:(afi-list)
|   +---rw afi-list
|       +---rw address-list*    simple-address
+---:(instance-id)
|   +---rw instance-id
|       +---rw instance-id?    instance-id-type
|       +---rw mask-length?    uint8
|       +---rw address?        simple-address
+---:(as-number-lcaf)
|   +---rw as-number-lcaf
|       +---rw as?            inet:as-number
|       +---rw address?        simple-address
+---:(application-data)
|   +---rw application-data
|       +---rw address?
|           |
|           |   simple-address
|       +---rw protocol?            uint8
|       +---rw ip-tos?              int32
|       +---rw local-port-low?
|           |
|           |   inet:port-number
|       +---rw local-port-high?
|           |
|           |   inet:port-number
|       +---rw remote-port-low?
|           |
|           |   inet:port-number
|       +---rw remote-port-high?
|           |
|           |   inet:port-number
+---:(geo-coordinates)
|   +---rw geo-coordinates
|       +---rw latitude?            bits
|       +---rw latitude-degrees?    uint8
|       +---rw latitude-minutes?    uint8
|       +---rw latitude-seconds?    uint8
|       +---rw longitude?           bits
|       +---rw longitude-degrees?    uint16
|       +---rw longitude-minutes?    uint8
|       +---rw longitude-seconds?    uint8
|       +---rw altitude?            int32
|       +---rw address?

```



```

|                                     simple-address
+---:(nat-traversal)
|   +---rw nat-traversal
|       +---rw ms-udp-port?          uint16
|       +---rw etr-udp-port?        uint16
|       +---rw global-etr-rloc?
|           |
|           |   simple-address
|       +---rw ms-rloc?
|           |
|           |   simple-address
|       +---rw private-etr-rloc?
|           |
|           |   simple-address
|       +---rw rtr-rlocs*
|           |
|           |   simple-address
+---:(explicit-locator-path)
|   +---rw explicit-locator-path
|       +---rw hop* [hop-id]
|           +---rw hop-id          string
|           +---rw address?        simple-address
|           +---rw lrs-bits?       bits
+---:(source-dest-key)
|   +---rw source-dest-key
|       +---rw source?            simple-address
|       +---rw dest?              simple-address
+---:(key-value-address)
|   +---rw key-value-address
|       +---rw key?                simple-address
|       +---rw value?              simple-address
+---:(service-path)
|   +---rw service-path
|       +---rw service-path-id?
|           |
|           |   service-path-id-type
|       +---rw service-index?      uint8
+---rw rlocs?                      leafref
|   -> /rt:routing/control-plane-protocols
|       /control-plane-protocol/lisp:lisp
|       /locator-sets
|       /locator-set/locator-set-name
+---rw record-ttl?                  uint32
+---rw want-map-notify?             boolean
+---rw proxy-reply?                 boolean
+---rw registration-interval?       uint16

```

4.2. Module Definition

```

<CODE BEGINS> file "ietf-lisp-etr@2019-02-23.yang"
module ietf-lisp-etr {
  yang-version 1.1;

```

```
namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-etr";

prefix lisp-etr;

// RFC Ed.: replace occurrences of XXXX with actual RFC number
// and remove this note
import ietf-lisp {
  prefix lisp;
  reference "RFC XXXX: LISP YANG model";
}
import ietf-lisp-address-types {
  prefix lcaf;
  reference "RFC XXXX: LISP YANG model";
}
import ietf-inet-types {
  prefix inet;
  reference "RFC 6991: Common YANG Data Types";
}
import ietf-routing {
  prefix "rt";
  reference
    "RFC 8349: A YANG Data Model for Routing Management
    (NMDA version)";
}

organization
  "IETF LISP (Locator/ID Separation Protocol) Working Group";
contact
  "WG Web:    <http://tools.ietf.org/wg/lisp/>
  WG List:    <mailto:lisp@ietf.org>

  Editor:     Vina Ermagan
               <mailto:ermagan@gmail.com>

  Editor:     Alberto Rodriguez-Natal
               <mailto:natal@cisco.com>

  Editor:     Reshad Rahman
               <mailto:rrahman@cisco.com>";
description
  "This YANG module defines the generic parameters for a LISP
  ETR. The module can be extended by vendors to define
  vendor-specific parameters and policies.

  Copyright (c) 2018 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
```

without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.

";

reference "RFC XXXX";

revision 2019-02-23 {

description

"Initial revision.";

reference

"<https://tools.ietf.org/html/rfc6830>";

}

augment "/rt:routing/rt:control-plane-protocols"

+ "/rt:control-plane-protocol/lisp:lisp" {

when "lisp:lisp-role/lisp:lisp-role-type = 'lisp:etr' or
lisp:lisp-role/lisp:lisp-role-type = 'lisp:petr'" {

description

"Augment is valid when LISP device type is (P)ETR.";

}

description

"This augments LISP devices list with (P)ETR specific
parameters.";

container etr {

presence "LISP (P)ETR operation enabled";

description

"(P)ETR parameters.";

container map-servers {

when "../lisp:lisp-role/lisp:lisp-role-type = 'lisp:etr'" {

description

"Container exists only when LISP device type is ETR.";

}

description

"Map-Servers that the ETR uses.";

list map-server {

key "ms-address";

description

"Each Map-Server within the list of Map-Servers.";

leaf ms-address {

type inet:ip-address;

description

"Map-Server address.";

```
    }
    uses lisp:auth-key;
  }
}

container local-eids {
  when "../../../lisp:lisp-role/lisp:lisp-role-type = 'lisp:etr'" {
    description
      "Container exists only when LISP device type is ETR.";
  }
  description
    "VPNs served by the ETR.";
  list vpn {
    key "instance-id";
    description
      "VPN for local-EIDs.";
    leaf instance-id {
      type leafref {
        path "/rt:routing/rt:control-plane-protocols"
          + "/rt:control-plane-protocol/lisp:lisp"
          + "/lisp:vpns/lisp:vpn"
          + "/lisp:instance-id";
      }
      description
        "VPN identifier.";
    }
  }
  container eids {
    description
      "EIDs served by the ETR.";
    list local-eid {
      key "id";
      description
        "List of local EIDs.";
      leaf id {
        type lisp:eid-id;
        description
          "Unique id of local EID.";
      }
    }
    container eid-address {
      uses lcaf:lisp-address;
      description
        "EID address in generic LISP address format.";
    }
    leaf rlocs {
      type leafref {
        path "/rt:routing/rt:control-plane-protocols"
          + "/rt:control-plane-protocol/lisp:lisp"
          + "/lisp:locator-sets/lisp:locator-set"
      }
    }
  }
}
```

```

        + "/lisp:locator-set-name";
    }
    description
        "Locator set mapped to this local EID.";
}
leaf record-ttl {
    type uint32;
    units minutes;
    description
        "Validity period of the EID to RLOCs mapping provided
        in Map-Replies.";
}
leaf want-map-notify {
    type boolean;
    default "true";
    description
        "Flag which if set in a Map-Register requests that a
        Map-Notify be sent in response.";
}
leaf proxy-reply {
    type boolean;
    default "false";
    description
        "Flag which if set in a Map-Register requests that the
        Map-Server proxy Map-Replies for the ETR.";
}
leaf registration-interval {
    type uint16;
    units "seconds";
    default "60";
    description
        "Interval between consecutive Map-Register messages.";
}
}
}
}
}
}
}
<CODE ENDS>
```

5. LISP-Map-Server Module

This module captures the configuration data model of a LISP Map Server [RFC6833]. The model also captures some operational data elements.

5.1. Module Structure

```

module: ietf-lisp-mapserver
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/lisp:lisp:
  +--rw map-server!
    +--rw sites
      +--rw site* [site-id]
        +--rw site-id          uint64
        +--rw authentication-keys
          +--rw authentication-key* [auth-key-id]
            +--rw auth-key-id      string
            +--rw auth-algorithm-id*
              | lisp:auth-algorithm-type
            +--rw auth-key-value?  string
        +--rw xtr-ids* [xtr-id]
          +--rw xtr-id            uint64
          +--rw authentication-keys
            +--rw authentication-key* [auth-key-id]
              +--rw auth-key-id      string
              +--rw auth-algorithm-id*
                | lisp:auth-algorithm-type
              +--rw auth-key-value?  string
    +--rw vpns
      +--rw vpn* [instance-id]
        +--rw instance-id      lcaf:instance-id-type
        +--rw mappings
          +--rw mapping* [eid-id]
            +--rw eid-id          lisp:eid-id
            +--rw eid-address
              +--rw address-type
                | lisp-address-family-ref
              +--rw (address)?
                +--:(no-address)
                | +--rw no-address?          empty
                +--:(ipv4)
                | +--rw ipv4?
                  | inet:ipv4-address
                +--:(ipv4-prefix)
                | +--rw ipv4-prefix?
                  | inet:ipv4-prefix
                +--:(ipv6)
                | +--rw ipv6?
                  | inet:ipv6-address
                +--:(ipv6-prefix)
                | +--rw ipv6-prefix?
                  | inet:ipv6-prefix
                +--:(mac)

```

```

|   +--rw mac?
|       yang:mac-address
+---:(distinguished-name)
|   +--rw distinguished-name?
|       distinguished-name-type
+---:(as-number)
|   +--rw as-number?
|       inet:as-number
+---:(null-address)
|   +--rw null-address
|       +--rw address?    empty
+---:(afi-list)
|   +--rw afi-list
|       +--rw address-list*  simple-address
+---:(instance-id)
|   +--rw instance-id
|       +--rw instance-id?   instance-id-type
|       +--rw mask-length?   uint8
|       +--rw address?       simple-address
+---:(as-number-lcaf)
|   +--rw as-number-lcaf
|       +--rw as?            inet:as-number
|       +--rw address?       simple-address
+---:(application-data)
|   +--rw application-data
|       +--rw address?
|           |
|           |   simple-address
+--rw protocol?          uint8
+--rw ip-tos?            int32
+--rw local-port-low?
|   |
|   |   inet:port-number
+--rw local-port-high?
|   |
|   |   inet:port-number
+--rw remote-port-low?
|   |
|   |   inet:port-number
+--rw remote-port-high?
|       |
|       |   inet:port-number
+---:(geo-coordinates)
|   +--rw geo-coordinates
|       +--rw latitude?      bits
|       +--rw latitude-degrees?  uint8
|       +--rw latitude-minutes?  uint8
|       +--rw latitude-seconds?  uint8
|       +--rw longitude?       bits
|       +--rw longitude-degrees? uint16
|       +--rw longitude-minutes? uint8
|       +--rw longitude-seconds? uint8
|       +--rw altitude?        int32

```

```

    +--rw address?
        |
        | simple-address
    +--:(nat-traversal)
        |
        | +--rw nat-traversal
        |   |
        |   | +--rw ms-udp-port?          uint16
        |   | +--rw etr-udp-port?        uint16
        |   | +--rw global-etr-rloc?
        |   |   |
        |   |   | simple-address
        |   | +--rw ms-rloc?
        |   |   |
        |   |   | simple-address
        |   | +--rw private-etr-rloc?
        |   |   |
        |   |   | simple-address
        |   | +--rw rtr-rlocs*
        |   |   |
        |   |   | simple-address
    +--:(explicit-locator-path)
        |
        | +--rw explicit-locator-path
        |   |
        |   | +--rw hop* [hop-id]
        |   |   |
        |   |   | +--rw hop-id          string
        |   |   | +--rw address?        simple-address
        |   |   | +--rw lrs-bits?       bits
    +--:(source-dest-key)
        |
        | +--rw source-dest-key
        |   |
        |   | +--rw source?              simple-address
        |   | +--rw dest?                simple-address
    +--:(key-value-address)
        |
        | +--rw key-value-address
        |   |
        |   | +--rw key?                  simple-address
        |   | +--rw value?                simple-address
    +--:(service-path)
        |
        | +--rw service-path
        |   |
        |   | +--rw service-path-id?
        |   |   |
        |   |   | service-path-id-type
        |   | +--rw service-index?        uint8
    +--rw site-id*                          uint64
    +--rw more-specifics-accepted?          boolean
    +--rw mapping-expiration-timeout?       int16
    +--ro first-registration-time?
        |
        | yang:date-and-time
    +--ro last-registration-time?
        |
        | yang:date-and-time
    +--rw mapping-records
        |
        | +--rw mapping-record* [xtr-id]
        |   |
        |   | +--rw xtr-id
        |   |   |
        |   |   | lisp:xtr-id-type
        |   | +--rw site-id?              uint64
        |   | +--rw eid
        |   |   |
        |   |   | +--rw address-type
        |   |   |   |
        |   |   |   | lisp-address-family-ref

```



```

+--rw (address)?
+--:(no-address)
|   +--rw no-address?
|       empty
+--:(ipv4)
|   +--rw ipv4?
|       inet:ipv4-address
+--:(ipv4-prefix)
|   +--rw ipv4-prefix?
|       inet:ipv4-prefix
+--:(ipv6)
|   +--rw ipv6?
|       inet:ipv6-address
+--:(ipv6-prefix)
|   +--rw ipv6-prefix?
|       inet:ipv6-prefix
+--:(mac)
|   +--rw mac?
|       yang:mac-address
+--:(distinguished-name)
|   +--rw distinguished-name?
|       distinguished-name-type
+--:(as-number)
|   +--rw as-number?
|       inet:as-number
+--:(null-address)
|   +--rw null-address
|   +--rw address?    empty
+--:(afi-list)
|   +--rw afi-list
|   +--rw address-list*
|       simple-address
+--:(instance-id)
|   +--rw instance-id
|   +--rw instance-id?
|       |   instance-id-type
|   +--rw mask-length?    uint8
|   +--rw address?
|       simple-address
+--:(as-number-lcaf)
|   +--rw as-number-lcaf
|   +--rw as?            inet:as-number
|   +--rw address?      simple-address
+--:(application-data)
|   +--rw application-data
|   +--rw address?
|       |   simple-address
|   +--rw protocol?      uint8

```

```

+--rw ip-tos? int32
+--rw local-port-low?
|   inet:port-number
+--rw local-port-high?
|   inet:port-number
+--rw remote-port-low?
|   inet:port-number
+--rw remote-port-high?
|   inet:port-number
+--:(geo-coordinates)
+--rw geo-coordinates
+--rw latitude? bits
+--rw latitude-degrees? uint8
+--rw latitude-minutes? uint8
+--rw latitude-seconds? uint8
+--rw longitude? bits
+--rw longitude-degrees?
|   uint16
+--rw longitude-minutes? uint8
+--rw longitude-seconds? uint8
+--rw altitude? int32
+--rw address?
|   simple-address
+--:(nat-traversal)
+--rw nat-traversal
+--rw ms-udp-port? uint16
+--rw etr-udp-port? uint16
+--rw global-etr-rloc?
|   simple-address
+--rw ms-rloc?
|   simple-address
+--rw private-etr-rloc?
|   simple-address
+--rw rtr-rlocs*
|   simple-address
+--:(explicit-locator-path)
+--rw explicit-locator-path
+--rw hop* [hop-id]
|   +--rw hop-id string
|   +--rw address?
|   |   simple-address
+--rw lrs-bits? bits
+--:(source-dest-key)
+--rw source-dest-key
|   +--rw source? simple-address
|   +--rw dest? simple-address
+--:(key-value-address)
+--rw key-value-address

```

```

|         +--rw key?          simple-address
|         +--rw value?       simple-address
+---:(service-path)
|         +--rw service-path
|         +--rw service-path-id?
|         |         service-path-id-type
|         +--rw service-index?      uint8
+--rw time-to-live?                  uint32
+--ro creation-time?
|         yang:date-and-time
+--rw authoritative?                 bits
+--rw static?                         boolean
+--rw (locator-list)?
+---:(negative-mapping)
|         +--rw map-reply-action?
|         |         map-reply-action
+---:(positive-mapping)
|         +--rw rlocs
|         +--rw locator* [id]
|         |         +--rw id
|         |         |         string
|         +--rw locator-address
|         |         +--rw address-type
|         |         |         lisp-address-family-ref
|         +--rw (address)?
|         |         +---:(no-address)
|         |         |         +--rw no-address?
|         |         |         |         empty
|         |         +---:(ipv4)
|         |         |         +--rw ipv4?
|         |         |         |         inet:ipv4-address
|         |         +---:(ipv4-prefix)
|         |         |         +--rw ipv4-prefix?
|         |         |         |         inet:ipv4-prefix
|         |         +---:(ipv6)
|         |         |         +--rw ipv6?
|         |         |         |         inet:ipv6-address
|         |         +---:(ipv6-prefix)
|         |         |         +--rw ipv6-prefix?
|         |         |         |         inet:ipv6-prefix
|         |         +---:(mac)
|         |         |         +--rw mac?
|         |         |         |         yang:mac-address
|         |         +---:(distinguished-name)
|         |         |         +--rw distinguished-name?
|         |         |         |         distinguished-name-type
|         |         +---:(as-number)
|         |         |         +--rw as-number?

```

```

|             inet:as-number
+--:(null-address)
|   +--rw null-address
|   +--rw address?
|       empty
+--:(afi-list)
|   +--rw afi-list
|   +--rw address-list*
|       simple-address
+--:(instance-id)
|   +--rw instance-id
|   +--rw instance-id?
|       | instance-id-type
+--rw mask-length?
|   uint8
+--rw address?
|       simple-address
+--:(as-number-lcaf)
|   +--rw as-number-lcaf
|   +--rw as?
|       | inet:as-number
+--rw address?
|       simple-address
+--:(application-data)
|   +--rw application-data
|   +--rw address?
|       | simple-address
+--rw protocol?
|   uint8
+--rw ip-tos?
|   int32
+--rw local-port-low?
|   inet:port-number
+--rw local-port-high?
|   inet:port-number
+--rw remote-port-low?
|   inet:port-number
+--rw remote-port-high?
|   inet:port-number
+--:(geo-coordinates)
|   +--rw geo-coordinates
|   +--rw latitude?
|       | bits
+--rw latitude-degrees?
|   uint8
+--rw latitude-minutes?
|   uint8
+--rw latitude-seconds?

```

```

|         uint8
+--rw longitude?
|         bits
+--rw longitude-degrees?
|         uint16
+--rw longitude-minutes?
|         uint8
+--rw longitude-seconds?
|         uint8
+--rw altitude?
|         int32
+--rw address?
|         simple-address
+--:(nat-traversal)
+--rw nat-traversal
+--rw ms-udp-port?
|         uint16
+--rw etr-udp-port?
|         uint16
+--rw global-etr-rloc?
|         simple-address
+--rw ms-rloc?
|         simple-address
+--rw private-etr-rloc?
|         simple-address
+--rw rtr-rlocs*
|         simple-address
+--:(explicit-locator-path)
+--rw explicit-locator-path
+--rw hop* [hop-id]
|         hop-id
|         string
+--rw address?
|         simple-address
+--rw lrs-bits?
|         bits
+--:(source-dest-key)
+--rw source-dest-key
+--rw source?
|         simple-address
+--rw dest?
|         simple-address
+--:(key-value-address)
+--rw key-value-address
+--rw key?
|         simple-address
+--rw value?
|         simple-address

```

```

+--:(service-path)
+--rw service-path
+--rw service-path-id?
|   service-path-id-type
+--rw service-index?
|   uint8
+--rw priority?
|   uint8
+--rw weight?
|   uint8
+--rw multicast-priority?
|   uint8
+--rw multicast-weight?
|   uint8
+--ro counters
+--ro map-registers-in?          yang:counter64
+--ro map-registers-in-auth-failed? yang:counter64
+--ro map-notify-records-out?     yang:counter64
+--ro proxy-reply-records-out?    yang:counter64
+--ro map-requests-forwarded-out? yang:counter64
+--rw mapping-system-type?      lisp:mapping-system-ref
+--ro summary
+--ro number-configured-sites?   uint32
+--ro number-registered-sites?   uint32
+--ro af-datum
+--ro af-data* [address-type]
+--ro address-type
|   lcaf:lisp-address-family-ref
+--ro number-configured-eids?    uint32
+--ro number-registered-eids?    uint32
+--ro counters
+--ro map-registers-in?          yang:counter64
+--ro map-registers-in-auth-failed? yang:counter64
+--ro map-notify-records-out?     yang:counter64
+--ro proxy-reply-records-out?    yang:counter64
+--ro map-requests-forwarded-out? yang:counter64

```

5.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp-mapserver@2019-03-05.yang"
module ietf-lisp-mapserver {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver";

  prefix lisp-ms;

  // RFC Ed.: replace occurrences of XXXX with actual RFC number
```

```
// and remove this note
import ietf-lisp {
  prefix lisp;
  reference "RFC XXXX: LISP YANG model";
}
import ietf-lisp-address-types {
  prefix lcaf;
  reference "RFC XXXX: LISP YANG model";
}
import ietf-yang-types {
  prefix yang;
  reference "RFC 6991: Common YANG Data Types";
}
import ietf-routing {
  prefix "rt";
  reference
    "RFC 8349: A YANG Data Model for Routing Management
    (NMDA version)";
}

organization
  "IETF LISP (Locator/ID Separation Protocol) Working Group";
contact
  "WG Web:    <http://tools.ietf.org/wg/lisp/>
  WG List:    <mailto:lisp@ietf.org>

  Editor:     Vina Ermagan
              <mailto:ermagan@gmail.com>

  Editor:     Alberto Rodriguez-Natal
              <mailto:natal@cisco.com>

  Editor:     Reshad Rahman
              <mailto:rrahman@cisco.com>";
description
  "This YANG module defines the generic parameters for a LISP
  Map-Server. The module can be extended by vendors to define
  vendor-specific parameters and policies.

  Copyright (c) 2018 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).
```

```
    This version of this YANG module is part of RFC XXXX; see
    the RFC itself for full legal notices.
";

reference "RFC XXXX";

revision 2019-03-05 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6833";
}

identity ms {
  base lisp:lisp-role;
  description
    "LISP Map-Server.";
}

grouping ms-counters {
  description "Grouping that defines map-server counters.";
  container counters {
    config false;
    description "Container for the counters";

    leaf map-registers-in {
      type yang:counter64;
      description "Number of incoming Map-Register messages";
    }

    leaf map-registers-in-auth-failed {
      type yang:counter64;
      description
        "Number of incoming Map-Register messages failed
        authentication";
    }

    leaf map-notify-records-out {
      type yang:counter64;
      description
        "Number of outgoing Map-Notify records";
    }

    leaf proxy-reply-records-out {
      type yang:counter64;
      description
        "Number of outgoing proxy Map-Reply records";
    }
  }
}
```



```
    leaf map-requests-forwarded-out {
      type yang:counter64;
      description
        "Number of outgoing Map-Requests forwarded to ETR";
    }
  }
}

augment "/rt:routing/rt:control-plane-protocols"
+ "/rt:control-plane-protocol/lisp:lisp" {
  when "lisp:lisp-role/lisp:lisp-role-type = 'lisp-ms:ms'" {
    description
      "Augment is valid when LISP device type is Map-Server.";
  }
  description
    "This augments LISP devices list with Map-Server specific
    parameters.";
  container map-server {
    presence "LISP Map-Server operation enabled";
    description
      "Map-Server parameters.";
    container sites{
      description
        "Sites to accept registrations from.";
      list site {
        key site-id;
        description
          "Site that can send registrations.";
        leaf site-id {
          type uint64;
          description "Site ID";
        }
        uses lisp:auth-key;
        list xtr-ids {
          key xtr-id;
          description "xTR-ID specific configuration.";
          leaf xtr-id {
            type uint64;
            description "xTR ID";
          }
          uses lisp:auth-key;
        }
      }
    }
  }
  container vpns {
    description
      "VPNs for which the Map-Server accepts registrations.";
    list vpn {
```

```
key "instance-id";
description
  "VPN instances in the Map-Server.";
leaf instance-id {
  type lcaf:instance-id-type;
  description
    "VPN identifier.";
}
container mappings {
  description
    "EIDs registered by device.";
  list mapping {
    key "eid-id";
    description
      "List of EIDs registered by device.";
    leaf eid-id {
      type lisp:eid-id;
      description
        "Id of the EID registered.";
    }
    container eid-address {
      uses lcaf:lisp-address;
      description
        "EID in generic LISP address format registered
        with the Map-Server.";
    }
    leaf-list site-id {
      type uint64;
      description "Site ID";
    }
    leaf more-specifics-accepted {
      type boolean;
      default "false";
      description
        "Flag indicating if more specific prefixes
        can be registered.";
    }
    leaf mapping-expiration-timeout {
      type int16;
      units "seconds";
      default "180"; //3 times the mapregister int
      description
        "Time before mapping is expired if no new
        registrations are received.";
    }
    leaf first-registration-time {
      type yang:date-and-time;
      config false;
    }
  }
}
```

```
        description
            "Time at which the first registration for this EID
            was received";
    }
    leaf last-registration-time {
        type yang:date-and-time;
        config false;
        description
            "Time at which the last registration for this EID
            was received";
    }
    container mapping-records {
        description
            "Datastore of registered mappings.";
        list mapping-record {
            key xtr-id;
            description
                "Registered mapping.";
            leaf xtr-id {
                type lisp:xtr-id-type;
                description "xTR ID";
            }
            leaf site-id {
                type uint64;
                description "Site ID";
            }
        }
        uses lisp:mapping;
    }
}
}
uses ms-counters;
}
leaf mapping-system-type {
    type lisp:mapping-system-ref;
    description
        "A reference to the mapping system";
}

container summary {
    config false;
    description "Summary state information";

    leaf number-configured-sites {
        type uint32;
        description "Number of configured LISP sites";
    }
}
```

```

    leaf number-registered-sites {
        type uint32;
        description "Number of registered LISP sites";
    }
    container af-datum {
        description "Number of configured EIDs per each AF";

        list af-data {
            key "address-type";
            description "Number of configured EIDs for this AF";
            leaf address-type {
                type lcaf:lisp-address-family-ref;
                description "AF type";
            }
            leaf number-configured-eids {
                type uint32;
                description "Number of configured EIDs for this AF";
            }
            leaf number-registered-eids {
                type uint32;
                description "Number of registered EIDs for this AF";
            }
        }
    }
}
uses ms-counters;
}
}
<CODE ENDS>

```

6. LISP-Map-Resolver Module

This module captures the configuration data model of a LISP Map Resolver [RFC6833]. The model also captures some operational data elements.

6.1. Module Structure

```

module: ietf-lisp-mapresolver
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/lisp:lisp:
    +--rw map-resolver!
      +--rw mapping-system-type?   lisp:mapping-system-ref
      +--rw ms-address?            inet:ip-address

```

6.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp-mapresolver@2019-02-23.yang"
module ietf-lisp-mapresolver {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-mapresolver";

  prefix lisp-mr;

  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note
  import ietf-lisp {
    prefix lisp;
    reference "RFC XXXX: LISP YANG model";
  }
  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management
       (NMDA version)";
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/lisp/>
    WG List:  <mailto:lisp@ietf.org>

    Editor:   Vina Ermagan
              <mailto:ermagan@gmail.com>

    Editor:   Alberto Rodriguez-Natal
              <mailto:natal@cisco.com>

    Editor:   Reshad Rahman
              <mailto:rrahman@cisco.com>";
  description
    "This YANG module defines the generic parameters for a LISP
    Map-Resolver. The module can be extended by vendors to define
    vendor-specific parameters and policies.

    Copyright (c) 2018 IETF Trust and the persons identified as
    authors of the code. All rights reserved."
```

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.
";

```
reference "RFC XXXX";

revision 2019-02-23 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6833";
}
identity mr {
  base lisp:lisp-role;
  description
    "LISP Map-Resolver.";
}

augment "/rt:routing/rt:control-plane-protocols"
  + "/rt:control-plane-protocol/lisp:lisp" {
  when "lisp:lisp-role/lisp:lisp-role-type = 'lisp-mr:mr'" {
    description
      "Augment is valid when LISP device type is Map-Resolver.";
  }
  description
    "This augments LISP devices list with Map-Resolver specific
    parameters.";
  container map-resolver {
    presence "LISP Map-Resolver operation enabled";
    description
      "Map-Resolver parameters.";
    leaf mapping-system-type {
      type lisp:mapping-system-ref;
      description
        "A reference to the mapping system";
    }
    leaf ms-address {
      when "../mapping-system-type='lisp:single-node-mapping-system'";
      type inet:ip-address;
      description
        "address to reach the Map Server when "
```

```
        + "lisp-mr:single-node-mapping-system is being used.";
    }
}
}
}
<CODE ENDS>
```

7. LISP-Address-Types Module

This module captures the various LISP address types, and is an essential building block used in other LISP modules.

7.1. Module Definition

```
<CODE BEGINS> file "ietf-lisp-address-types@2019-02-23.yang"
module ietf-lisp-address-types {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types";

  prefix laddr;

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types";
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/lisp/>
    WG List:  <mailto:lisp@ietf.org>

    Editor:   Vina Ermagan
              <mailto:ermagan@gmail.com>

    Editor:   Alberto Rodriguez-Natal
              <mailto:natal@cisco.com>

    Editor:   Reshad Rahman
              <mailto:rrahman@cisco.com>";
  description
    "This YANG module defines the LISP Canonical Address Formats
    (LCAF) for LISP. The module can be extended by vendors to
```

define vendor-specific parameters.

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.

```
    ";
// RFC Ed.: replace XXXX with actual RFC number and remove
// this note
reference "RFC XXXX";

revision 2019-02-23 {
  description
    "Initial revision.";
  reference
    "RC8060: LISP Canonical Address Format (LCAF)";
}
identity lisp-address-family {
  description
    "Base identity from which identities describing LISP address
    families are derived.";
}
identity no-address-afi {
  base lisp-address-family;
  description
    "IANA Reserved.";
}
identity ipv4-afi {
  base lisp-address-family;
  description
    "IANA IPv4 address family.";
}
identity ipv4-prefix-afi {
  base lisp-address-family;
  description
    "IANA IPv4 address family prefix.";
}
identity ipv6-afi {
  base lisp-address-family;
```



```
    description
      "IANA IPv6 address family.";
  }
  identity ipv6-prefix-afi {
    base lisp-address-family;
    description
      "IANA IPv6 address family prefix.";
  }
  identity mac-afi {
    base lisp-address-family;
    description
      "IANA MAC address family.";
  }
  identity distinguished-name-afi {
    base lisp-address-family;
    description
      "IANA Distinguished Name address family.";
  }
  identity as-number-afi {
    base lisp-address-family;
    description
      "IANA AS Number address family.";
  }
  identity lcaf {
    base lisp-address-family;
    description
      "IANA LISP Canonical Address Format address family.";
  }
  identity null-address-lcaf {
    base lcaf;
    description
      "Null body LCAF type.";
  }
  identity afi-list-lcaf {
    base lcaf;
    description
      "AFI-List LCAF type.";
  }
  identity instance-id-lcaf {
    base lcaf;
    description
      "Instance-ID LCAF type.";
  }
  identity as-number-lcaf {
    base lcaf;
    description
      "AS Number LCAF type.";
  }
}
```

```
identity application-data-lcaf {
  base lcaf;
  description
    "Application Data LCAF type.";
}
identity geo-coordinates-lcaf {
  base lcaf;
  description
    "Geo-coordinates LCAF type.";
}
identity opaque-key-lcaf {
  base lcaf;
  description
    "Opaque Key LCAF type.";
}
identity nat-traversal-lcaf {
  base lcaf;
  description
    "NAT-Traversal LCAF type.";
}
identity nonce-locator-lcaf {
  base lcaf;
  description
    "Nonce-Locator LCAF type.";
}
identity multicast-info-lcaf {
  base lcaf;
  description
    "Multicast Info LCAF type.";
}
identity explicit-locator-path-lcaf {
  base lcaf;
  description
    "Explicit Locator Path LCAF type.";
}
identity security-key-lcaf {
  base lcaf;
  description
    "Security Key LCAF type.";
}
identity source-dest-key-lcaf {
  base lcaf;
  description
    "Source/Dest LCAF type.";
}
identity replication-list-lcaf {
  base lcaf;
  description
```

```
        "Replication-List LCAF type.";
    }
    identity json-data-model-lcaf {
        base lcaf;
        description
            "JSON Data Model LCAF type.";
    }
    identity key-value-address-lcaf {
        base lcaf;
        description
            "Key/Value Address LCAF type.";
    }
    identity encapsulation-format-lcaf {
        base lcaf;
        description
            "Encapsulation Format LCAF type.";
    }
    identity service-path-lcaf {
        base lcaf;
        description
            "Service Path LCAF type.";
    }
    typedef instance-id-type {
        type uint32 {
            range "0..16777215";
        }
        description
            "Defines the range of values for an Instance ID.";
    }
    typedef service-path-id-type {
        type uint32 {
            range "0..16777215";
        }
        description
            "Defines the range of values for a Service Path ID.";
    }
    typedef distinguished-name-type {
        type string;
        description
            "Distinguished Name address.";
        reference
            "http://www.iana.org/assignments/address-family-numbers/
            address-family-numbers.xhtml";
    }
    typedef simple-address {
        type union {
            type inet:ip-address;
            type inet:ip-prefix;
        }
    }
```

```
    type yang:mac-address;
    type distinguished-name-type;
    type inet:as-number;
  }
  description
    "Union of address types that can be part of LCAFs.";
}

typedef lisp-address-family-ref {
  type identityref {
    base lisp-address-family;
  }
  description
    "LISP address family reference.";
}
typedef lcac-ref {
  type identityref {
    base lcac;
  }
  description
    "LCAF types reference.";
}

grouping lisp-address {
  description
    "Generic LISP address.";
  leaf address-type {
    type lisp-address-family-ref;
    mandatory true;
    description
      "Type of the LISP address.";
  }
  choice address {
    description
      "Various LISP address types, including IP, MAC, and LCAF.";

    leaf no-address {
      when "../address-type = 'laddr:no-address-afi'" {
        description
          "When AFI is 0.";
      }
      type empty;
      description
        "No address.";
    }
    leaf ipv4 {
      when "../address-type = 'laddr:ipv4-afi'" {
        description

```

```
        "When AFI is IPv4.";
    }
    type inet:ipv4-address;
    description
        "IPv4 address.";
}
leaf ipv4-prefix {
    when "../address-type = 'laddr:ipv4-prefix-afi'" {
        description
            "When AFI is IPv4.";
    }
    type inet:ipv4-prefix;
    description
        "IPv4 prefix.";
}
leaf ipv6 {
    when "../address-type = 'laddr:ipv6-afi'" {
        description
            "When AFI is IPv6.";
    }
    type inet:ipv6-address;
    description
        "IPv6 address.";
}
leaf ipv6-prefix {
    when "../address-type = 'laddr:ipv6-prefix-afi'" {
        description
            "When AFI is IPv6.";
    }
    type inet:ipv6-prefix;
    description
        "IPv6 address.";
}
leaf mac {
    when "../address-type = 'laddr:mac-afi'" {
        description
            "When AFI is MAC.";
    }
    type yang:mac-address;
    description
        "MAC address.";
}
leaf distinguished-name {
    when "../address-type = 'laddr:distinguished-name-afi'" {
        description
            "When AFI is distinguished-name.";
    }
    type distinguished-name-type;
}
```

```
    description
      "Distinguished Name address.";
  }
  leaf as-number {
    when "../address-type = 'laddr:as-number-afi'" {
      description
        "When AFI is as-number.";
    }
    type inet:as-number;
    description
      "AS Number.";
  }
  container null-address {
    when "../address-type = 'laddr:null-address-lcaf'" {
      description
        "When LCAF type is null.";
    }
    description
      "Null body LCAF type";
    leaf address {
      type empty;
      description
        "AFI address.";
    }
  }
  container afi-list {
    when "../address-type = 'laddr:afi-list-lcaf'" {
      description
        "When LCAF type is AFI-List.";
    }
    description
      "AFI-List LCAF type.";
    reference
      "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10#section-4.16.1";
    leaf-list address-list {
      type simple-address;
      description
        "List of AFI addresses.";
    }
  }
  container instance-id {
    when "../address-type = 'laddr:instance-id-lcaf'" {
      description
        "When LCAF type is Instance ID as per RFC8060.";
    }
    description
      "Instance ID LCAF type.";
```

```
reference
  "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
  #section-4.2";
leaf instance-id {
  type instance-id-type;
  description
    "Instance ID value.";
}
leaf mask-length {
  type uint8;
  description
    "Mask length.";
}
leaf address {
  type simple-address;
  description
    "AFI address.";
}
}
container as-number-lcaf {
  when "../address-type = 'laddr:as-number-lcaf'" {
    description
      "When LCAF type is AS-Number.";
  }
  description
    "AS Number LCAF type.";
  reference
    "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
    #section-4.3";
  leaf as {
    type inet:as-number;
    description
      "AS number.";
  }
  leaf address {
    type simple-address;
    description
      "AFI address.";
  }
}
container application-data {
  when "../address-type = 'laddr:application-data-lcaf'" {
    description
      "When LCAF type is Application Data.";
  }
  description
    "Application Data LCAF type.";
  reference
```

```
    "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
    #section-4.4";
  leaf address {
    type simple-address;
    description
      "AFI address.";
  }
  leaf protocol {
    type uint8;
    description
      "Protocol number.";
  }
  leaf ip-tos {
    type int32;
    description
      "Type of service field.";
  }
  leaf local-port-low {
    type inet:port-number;
    description
      "Low end of local port range.";
  }
  leaf local-port-high {
    type inet:port-number;
    description
      "High end of local port range.";
  }
  leaf remote-port-low {
    type inet:port-number;
    description
      "Low end of remote port range.";
  }
  leaf remote-port-high {
    type inet:port-number;
    description
      "High end of remote port range.";
  }
}
container geo-coordinates {
  when "../address-type = 'laddr:geo-coordinates-lcaf'" {
    description
      "When LCAF type is Geo-coordinates.";
  }
  description
    "Geo-coordinates LCAF type.";
  reference
    "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
    #section-4.5";
}
```



```
leaf latitude {
  type bits {
    bit N {
      description
        "Latitude bit.";
    }
  }
  description
    "Bit that selects between North and South latitude.";
}
leaf latitude-degrees {
  type uint8 {
    range "0 .. 90";
  }
  description
    "Degrees of latitude.";
}
leaf latitude-minutes {
  type uint8 {
    range "0..59";
  }
  description
    "Minutes of latitude.";
}
leaf latitude-seconds {
  type uint8 {
    range "0..59";
  }
  description
    "Seconds of latitude.";
}
leaf longitude {
  type bits {
    bit E {
      description
        "Longitude bit.";
    }
  }
  description
    "Bit that selects between East and West longitude.";
}
leaf longitude-degrees {
  type uint16 {
    range "0 .. 180";
  }
  description
    "Degrees of longitude.";
}
```

```
leaf longitude-minutes {
  type uint8 {
    range "0..59";
  }
  description
    "Minutes of longitude.";
}
leaf longitude-seconds {
  type uint8 {
    range "0..59";
  }
  description
    "Seconds of longitude.";
}
leaf altitude {
  type int32;
  description
    "Height relative to sea level in meters.";
}
leaf address {
  type simple-address;
  description
    "AFI address.";
}
}
container nat-traversal {
  when "../address-type = 'laddr:nat-traversal-lcaf'" {
    description
      "When LCAF type is NAT-Traversal.";
  }
  description
    "NAT-Traversal LCAF type.";
  reference
    "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
    #section-4.6";
  leaf ms-udp-port {
    type uint16;
    description
      "Map-Server UDP port (set to 4342).";
  }
  leaf etr-udp-port {
    type uint16;
    description
      "ETR UDP port.";
  }
  leaf global-etr-rloc {
    type simple-address;
    description
```

```
        "Global ETR RLOC address.";
    }
    leaf ms-rloc {
        type simple-address;
        description
            "Map-Server RLOC address.";
    }
    leaf private-etr-rloc {
        type simple-address;
        description
            "Private ETR RLOC address.";
    }
    leaf-list rtr-rlocs {
        type simple-address;
        description
            "List of RTR RLOC addresses.";
    }
}
container explicit-locator-path {
    when "../address-type = 'laddr:explicit-locator-path-lcaf'" {
        description
            "When LCAF type type is Explicit Locator Path.";
    }
    description
        "Explicit Locator Path LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
        #section-4.9";
    list hop {
        key "hop-id";
        ordered-by user;
        description
            "List of locator hops forming the explicit path.";
        leaf hop-id {
            type string {
                length "1..64";
            }
            description
                "Unique identifier for the hop.";
        }
        leaf address {
            type simple-address;
            description
                "AFI address.";
        }
        leaf lrs-bits {
            type bits {
                bit lookup {
```

```
        description
            "Lookup bit.";
    }
    bit rloc-probe {
        description
            "RLOC-probe bit.";
    }
    bit strict {
        description
            "Strict bit.";
    }
}
description
    "Flag bits per hop.";
}
}
container source-dest-key {
    when "../address-type = 'laddr:source-dest-key-lcaf'" {
        description
            "When LCAF type type is Source/Dest.";
    }
    description
        "Source/Dest LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
        #section-4.11";
    leaf source {
        type simple-address;
        description
            "Source address.";
    }
    leaf dest {
        type simple-address;
        description
            "Destination address.";
    }
}
container key-value-address {
    when "../address-type = 'laddr:key-value-address-lcaf'" {
        description
            "When LCAF type type is Key/Value Address.";
    }
    description
        "Key/Value Address LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
        #section-4.11";
```

```

leaf key {
  type simple-address;
  description
    "Address as Key.";
}
leaf value {
  type simple-address;
  description
    "Address as Value.";
}
}
container service-path {
  when "../address-type = 'laddr:service-path-lcaf'" {
    description
      "When LCAF type service path identifier.";
  }
  description
    "Service Path LCAF type.";
  reference
    "http://tools.ietf.org/html/draft-ermagan-lisp-nsh-00";
  leaf service-path-id {
    type service-path-id-type;
    description
      "Service path identifier for the path for NSH header";
  }
  leaf service-index {
    type uint8;
    description
      "Service path index for NSH header";
  }
}
}
}
}
<CODE ENDS>

```

7.2. Data Model examples

This section presents some simple and illustrative examples on how to configure LISP.

7.2.1. LISP protocol instance

The following is an example configuration for a LISP protocol instance with the name "LISP1". There are also 2 VNIs configured.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <network-instances
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
    <network-instance>
      <name>VRF-BLUE</name>
      <vrf-root/>
      <enabled>true</enabled>
    </network-instance>
    <network-instance>
      <name>VRF-RED</name>
      <vrf-root/>
      <enabled>true</enabled>
    </network-instance>
  </network-instances>
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
          lisp:lisp
        </type>
        <name>LISP1</name>
        <lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
          <lisp-role>
            <lisp-role-type>etr</lisp-role-type>
          </lisp-role>
          <lisp-role>
            <lisp-role-type>itr</lisp-role-type>
          </lisp-role>
          <vpns>
            <vpn>
              <instance-id>1000</instance-id>
              <iid-name>VRF-BLUE</iid-name>
            </vpn>
            <vpn>
              <instance-id>2000</instance-id>
              <iid-name>VRF-RED</iid-name>
            </vpn>
          </vpns>
        </lisp>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>
```

7.2.2. LISP ITR

The following is an example configuration for ITR functionality under "LISP1". There are 2 Map-Resolvers configured.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
          lisp:lisp
        </type>
        <name>LISP1</name>
        <lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
          <lisp-role>
            <lisp-role-type>itr</lisp-role-type>
          </lisp-role>
          <itr xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp-itr">
            <map-resolvers>
              <map-resolver>2001:db8:203:0:113::1</map-resolver>
              <map-resolver>2001:db8:204:0:113::1</map-resolver>
            </map-resolvers>
          </itr>
        </lisp>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>
```

7.2.3. LISP ETR

The following is an example configuration for ETR functionality under "LISP1". There are 2 Map-Servers and 2 local EIDs configured.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <network-instances
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
    <network-instance>
      <name>VRF-BLUE</name>
      <vrf-root/>
      <enabled>true</enabled>
    </network-instance>
```

```
<network-instance>
  <name>VRF-RED</name>
  <vrf-root/>
  <enabled>true</enabled>
</network-instance>
</network-instances>
<routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
  <control-plane-protocols>
    <control-plane-protocol>
      <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
        lisp:lisp
      </type>
      <name>LISP1</name>
      <lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
        <lisp-role>
          <lisp-role-type>etr</lisp-role-type>
        </lisp-role>
        <lisp-router-id>
          <site-id>1</site-id>
        </lisp-router-id>
        <vpns>
          <vpn>
            <instance-id>1000</instance-id>
            <iid-name>VRF-BLUE</iid-name>
          </vpn>
          <vpn>
            <instance-id>2000</instance-id>
            <iid-name>VRF-RED</iid-name>
          </vpn>
        </vpns>
      </lisp>
    </control-plane-protocol>
  </control-plane-protocols>
  <etr xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp-etr">
    <map-servers>
      <map-server>
        <ms-address>2001:db8:203:0:113::1</ms-address>
        <authentication-keys>
          <authentication-key>
            <auth-key-id>key1</auth-key-id>
            <auth-algorithm-id>
              hmac-sha-256-128
            </auth-algorithm-id>
            <auth-key-value>*Kye^$$1#gb91U04zpa</auth-key-value>
          </authentication-key>
        </authentication-keys>
      </map-server>
      <map-server>
        <ms-address>2001:db8:204:0:113::1</ms-address>
        <authentication-keys>
          <authentication-key>
```



```

        <auth-key-id>key1</auth-key-id>
        <auth-algorithm-id>
            hmac-sha-256-128
        </auth-algorithm-id>
        <auth-key-value>*Kye^$$1#gb91U04zpa</auth-key-value>
    </authentication-key>
</authentication-keys>
</map-server>
</map-servers>
<local-eids>
    <vpn>
        <instance-id>1000</instance-id>
        <eids>
            <local-eid>
                <id>2001:db8:400:0:100::0</id>
                <eid-address>
                    <address-type xmlns:laddr=
                        "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
                        laddr:ipv6-prefix-afi
                    </address-type>
                    <ipv6-prefix>2001:db8:400:0:100::/80</ipv6-prefix>
                </eid-address>
            </local-eid>
        </eids>
    </vpn>
    <vpn>
        <instance-id>2000</instance-id>
        <eids>
            <local-eid>
                <id>2001:db8:800:0:200::0</id>
                <eid-address>
                    <address-type xmlns:laddr=
                        "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
                        laddr:ipv6-prefix-afi
                    </address-type>
                    <ipv6-prefix>2001:db8:800:0:200::/80</ipv6-prefix>
                </eid-address>
            </local-eid>
        </eids>
    </vpn>
</local-eids>
</etr>
</lisp>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</config>

```

7.2.4. LISP Map-Server

The following is an example configuration for Map-Server functionality under "LISP1". There are 2 mappings configured.

```
<config xmlns="http://tail-f.com/ns/config/1.0">
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
          lisp:lisp
        </type>
        <name>LISP1</name>
        <lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
          <lisp-role>
            <lisp-role-type xmlns:lisp-ms=
              "urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver">
              lisp-ms:ms
            </lisp-role-type>
          </lisp-role>
          <map-server xmlns=
            "urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver">
            <sites>
              <site>
                <site-id>1</site-id>
                <authentication-keys>
                  <authentication-key>
                    <auth-key-id>key1</auth-key-id>
                    <auth-algorithm-id>
                      hmac-sha-256-128
                    </auth-algorithm-id>
                    <auth-key-value>*Kye^$$1#gb91U04zpa</auth-key-value>
                  </authentication-key>
                </authentication-keys>
              </site>
            </sites>
            <vpns>
              <vpn>
                <instance-id>1000</instance-id>
                <mappings>
                  <mapping>
                    <eid-id>1</eid-id>
                    <eid-address>
                      <address-type xmlns:laddr=
                        "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
                        laddr:ipv6-prefix-afi
                      </address-type>

```

```
        <ipv6-prefix>2001:db8:400:0:100::/80</ipv6-prefix>
      </eid-address>
    </mapping>
  </mappings>
</vpn>
<vpn>
  <instance-id>2000</instance-id>
  <mappings>
    <mapping>
      <eid-id>1</eid-id>
      <eid-address>
        <address-type xmlns:laddr=
"urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
          laddr:ipv6-prefix-afi
        </address-type>
        <ipv6-prefix>2001:db8:800:0:200::/80</ipv6-prefix>
      </eid-address>
    </mapping>
  </mappings>
</vpn>
</vpns>
</map-server>
</lisp>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</config>
```

8. Acknowledgments

The tree view and the YANG model shown in this document have been formatted with the 'pyang' tool.

9. IANA Considerations

The IANA is requested to assign a new namespace URI from the IETF XML registry.

This document registers the following namespace URIs in the IETF XML registry [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-lisp

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-itr

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-etr

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-mapresolver

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-address-types

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

10. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a pre-configured subset of all available NETCONF or RESTCONF protocol operations and content.

The security considerations of LISP control-plane [RFC6833] and LISP data-plane [RFC6830] as well as the LISP threat analysis [RFC7835] apply to this YANG model.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/  
lisp:lisp/
```

Access to the locator-sets node may modify which interfaces are used for data and/or control traffic as well as affect the load balancing of data-plane traffic. Access to the lisp-role node may prevent the device from perform its intended data-plane and/or control-plane operation. Access to the router-id node allows to modify the unique identifier of the device, which may result in disruption of its LISP control-plane operation. Access to the vpn node may allow to redirect data-plane traffic to erroneous local or remote network instances.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:map-server
```

Access to the sites node can prevent authorized devices from registering mappings in the Map-Server and/or allow unauthorized devices to so. Access to the vpn node can result in corrupted mapping state that may propagate across the LISP network, potentially resulting in forwarding of data-plane traffic to arbitrary destinations and general disruption of the data-plane operation. Access to mapping-system-type and/or ddt-mapping-system nodes may prevent the device to connect to the Mapping System infrastructure and consequentially to attract Map-Request messages.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:map-resolver
```

Access to mapping-system-type, ms-address and/or ddt-mapping-system nodes may prevent the device to connect to the Mapping System infrastructure and forward Map-Request messages.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:itr
```

Access to the rloc-probing node can increase the control-plane overhead in the device or affect the capability of the device to detect failures on the underlay. Access to the itr-rlocs node may prevent the device from getting Map-Reply messages. Access to the map-resolvers node can prevent the device from sending its Map-Request messages to valid Map-Resolvers. Access to the proxy-etr nodes can affect the capability of the device to send data-plane traffic towards non-LISP destinations. Access to the map-cache node can result in forwarding of data-plane traffic to arbitrary destinations and general disruption of data-plane operation.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:etr
```

Access to the map-servers node can prevent the device from registering its local mappings into the Mapping System. Access to the local-eids node can disrupt data-plane operation on the device and/or result in the device registering corrupted mappings into the Mapping System.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/  
lisp:lisp
```

Access to the locator-sets node can expose the locators the device is using for its control and/or data operation. Access to the lisp-role node can disclose the LISP roles instantiated at the device which facilitates mounting attacks against the device. Access to the router-id node can expose the unique identifier of device which may allow a third party to track its control-plane operation and/or impersonate the device. Access to the vpn node can leak the local mapping between LISP Instance IDs and local network instances.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:map-server
```

Access to the sites node can expose the credentials used to register mappings and allow unauthorized devices to do so. Access to the vpn node can expose the mappings currently registered in the device, which has privacy implications. Access to the mapping-system-type node may reveal the Mapping System in use which can be used to mount attacks against the device and/or the Mapping System. Access to the summary and counters nodes may expose operational statistics of the device.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:map-resolver
```

Access to the mapping-system-type node may reveal the Mapping System in use which can be used to mount attacks against the device and/or the Mapping System. Access to the ms-address and/or ddt-mapping-system nodes can leak the information about the Mapping System infrastructure used by the device, which can be used to block communication and/or mount attacks against it.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:itr
```

Access to the rloc-probing node can expose if and how the device is using control-plane signaling to probe underlay locators. Access to the itr-rlocs node may disclose the addresses the device is using to receive Map-Reply messages. Access to the map-resolvers node can expose the Map-Resolvers used by the device, which can be used to mount attacks against the device and/or the Mapping System. Access to the proxy-etrns node can disclose the PETRs used by the device, which can be used to mount attacks against the device and/or PETRs. Access to the map-cache node can expose the mappings currently cached in the device, which has privacy implications.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:etr
```

Access to the map-servers node can expose the credentials used by the device to register mappings into the Mapping System allowing an unauthorized device to impersonate and register mappings on behalf the authorized device. Access to the local-eids node can expose the local EIDs currently being served by the device, which has privacy implications.

11. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <<https://www.rfc-editor.org/info/rfc6832>>.
- [RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, DOI 10.17487/RFC6833, January 2013, <<https://www.rfc-editor.org/info/rfc6833>>.
- [RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, DOI 10.17487/RFC6836, January 2013, <<https://www.rfc-editor.org/info/rfc6836>>.

- [RFC7835] Saucez, D., Iannone, L., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Threat Analysis", RFC 7835, DOI 10.17487/RFC7835, April 2016, <<https://www.rfc-editor.org/info/rfc7835>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.
- [RFC8111] Fuller, V., Lewis, D., Ermagan, V., Jain, A., and A. Smirnov, "Locator/ID Separation Protocol Delegated Database Tree (LISP-DDT)", RFC 8111, DOI 10.17487/RFC8111, May 2017, <<https://www.rfc-editor.org/info/rfc8111>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

Authors' Addresses

Vina Ermagan
Google
USA

Email: ermagan@gmail.com

Alberto Rodriguez-Natal
Cisco Systems
San Jose, CA
USA

Email: natal@cisco.com

Florin Coras
Cisco Systems
San Jose, CA
USA

Email: fcoras@cisco.com

Carl Moberg
Cisco Systems
San Jose, CA
USA

Email: camoberg@cisco.com

Reshad Rahman
Cisco Systems
Canada

Email: rrahman@cisco.com

Albert Cabellos-Aparicio
Technical University of Catalonia
Barcelona
Spain

Email: acabello@ac.upc.edu

Fabio Maino
Cisco Systems
San Jose, CA
USA

Email: fmaino@cisco.com

LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: 28 August 2022

V. Ermagan
Google
A. Rodriguez-Natal
F. Coras
Cisco Systems
C. Moberg
Avassa
R. Rahman

A. Cabellos-Aparicio
Technical University of Catalonia
F. Maino
Cisco Systems
24 February 2022

LISP YANG Model
draft-ietf-lisp-yang-17

Abstract

This document describes a YANG data model to use with the Locator/ID Separation Protocol (LISP).

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 August 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
1.2. Tree Diagrams	3
1.3. Prefixes	3
2. LISP Module	4
2.1. Module Structure	4
2.2. Module Definition	7
3. LISP-ITR Module	19
3.1. Module Structure	19
3.2. Module Definition	24
4. LISP-ETR Module	28
4.1. Module Structure	28
4.2. Module Definition	30
5. LISP-Map-Server Module	34
5.1. Module Structure	34
5.2. Module Definition	43
6. LISP-Map-Resolver Module	49
6.1. Module Structure	50
6.2. Module Definition	50
7. LISP-Address-Types Module	52
7.1. Module Definition	52
7.2. Data Model examples	67
7.2.1. LISP protocol instance	67
7.2.2. LISP ITR	69
7.2.3. LISP ETR	69
7.2.4. LISP Map-Server	72
8. Acknowledgments	73
9. IANA Considerations	73
10. Security Considerations	76
11. Normative References	79
Authors' Addresses	82

1. Introduction

The Locator/ID Separation Protocol (LISP) defines several network elements subject to be configured. This document presents the YANG data models required for basic configuration of all major LISP [RFC6830] elements. The models also capture some essential operational data elements as well.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Tree Diagrams

This document uses the graphical representation of data models defined in [RFC8340].

1.3. Prefixes

The table below provides a summary of the prefixes used by this document.

Prefix	YANG module	Reference
lisp	ietf-lisp	Section 2
if	ietf-interfaces	[RFC8343]
lisp-at	ietf-lisp-address-types	Section 7
yang	ietf-yang-types	[RFC6991]
rt	ietf-routing	[RFC8022]
ni	ietf-network-instance	[RFC8529]
lisp-itr	ietf-lisp-itr	Section 3
inet	ietf-inet-types	[RFC6991]
lisp-etr	ietf-lisp-etr	Section 4
lisp-ms	ietf-lisp-mapserver	Section 5
lisp-mr	ietf-lisp-mapresolver	Section 6

Table 1: Prefixes and corresponding YANG modules

2. LISP Module

This is the base LISP module. It is further augmented by the LISP device role specific modules defined elsewhere in this document.

2.1. Module Structure

```

module: ietf-lisp
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol:
      +--rw lisp
        +--rw locator-sets
          +--rw locator-set* [locator-set-name]
            +--rw locator-set-name      string
            +--rw (locator-type)?
              +--:(local-interface)
                +--rw interface* [interface-ref]
                  +--rw interface-ref    if:interface-ref
                  +--rw priority?         uint8
                  +--rw weight?           uint8

```

```

|      +---rw multicast-priority?    uint8
|      +---rw multicast-weight?      uint8
+---:(general-locator)
|      +---rw locator* [locator-id]
|      +---rw locator-id             string
|      +---rw locator-address
|      |      +---rw address-type
|      |      |      lisp-address-family-ref
|      +---rw (address)?
|      |      +---:(no-address)
|      |      |      +---rw no-address?          empty
|      +---:(ipv4)
|      |      +---rw ipv4?
|      |      |      inet:ipv4-address
|      +---:(ipv4-prefix)
|      |      +---rw ipv4-prefix?
|      |      |      inet:ipv4-prefix
|      +---:(ipv6)
|      |      +---rw ipv6?
|      |      |      inet:ipv6-address
|      +---:(ipv6-prefix)
|      |      +---rw ipv6-prefix?
|      |      |      inet:ipv6-prefix
|      +---:(mac)
|      |      +---rw mac?
|      |      |      yang:mac-address
|      +---:(distinguished-name)
|      |      +---rw distinguished-name?
|      |      |      distinguished-name-type
|      +---:(as-number)
|      |      +---rw as-number?
|      |      |      inet:as-number
|      +---:(null-address)
|      |      +---rw null-address
|      |      |      +---rw address?          empty
|      +---:(afi-list)
|      |      +---rw afi-list
|      |      |      +---rw address-list*
|      |      |      |      simple-address
|      +---:(instance-id)
|      |      +---rw instance-id
|      |      |      +---rw instance-id?
|      |      |      |      instance-id-type
|      |      +---rw mask-length?    uint8
|      |      +---rw address?        simple-address
|      +---:(as-number-lcaf)
|      |      +---rw as-number-lcaf
|      |      |      +---rw as?            inet:as-number

```

```

|         +---rw address?    simple-address
+---:(application-data)
|   +---rw application-data
|     +---rw address?
|       | simple-address
+---rw protocol?          uint8
+---rw ip-tos?            int32
+---rw local-port-low?
|   | inet:port-number
+---rw local-port-high?
|   | inet:port-number
+---rw remote-port-low?
|   | inet:port-number
+---rw remote-port-high?
|   | inet:port-number
+---:(geo-coordinates)
|   +---rw geo-coordinates
|     +---rw latitude?      bits
|     +---rw latitude-degrees?  uint8
|     +---rw latitude-minutes?  uint8
|     +---rw latitude-seconds?  uint8
|     +---rw longitude?      bits
|     +---rw longitude-degrees? uint16
|     +---rw longitude-minutes? uint8
|     +---rw longitude-seconds? uint8
|     +---rw altitude?      int32
|     +---rw address?
|       | simple-address
+---:(nat-traversal)
|   +---rw nat-traversal
|     +---rw ms-udp-port?    uint16
|     +---rw etr-udp-port?   uint16
|     +---rw global-etr-rloc?
|       | simple-address
+---rw ms-rloc?
|   | simple-address
+---rw private-etr-rloc?
|   | simple-address
+---rw rtr-rlocs*
|   | simple-address
+---:(explicit-locator-path)
|   +---rw explicit-locator-path
|     +---rw hop* [hop-id]
|       +---rw hop-id      string
|       +---rw address?    simple-address
|       +---rw lrs-bits?   bits
+---:(source-dest-key)
|   +---rw source-dest-key

```



```

|                                     | +--rw source?      simple-address
|                                     | +--rw dest?        simple-address
|                                     | +---:(key-value-address)
|                                     |   +--rw key-value-address
|                                     |   +--rw key?         simple-address
|                                     |   +--rw value?       simple-address
|                                     | +---:(service-path)
|                                     |   +--rw service-path
|                                     |   +--rw service-path-id?
|                                     |     |
|                                     |     | service-path-id-type
|                                     |   +--rw service-index?    uint8
|                                     |
+---rw priority?                uint8
+---rw weight?                  uint8
+---rw multicast-priority?      uint8
+---rw multicast-weight?        uint8
+---rw lisp-role* [lisp-role-type]
|   +--rw lisp-role-type      lisp-role-ref
+---rw lisp-router-id
|   +--rw site-id?           uint64
|   +--rw xtr-id?            lisp:xtr-id-type
+---rw vpns
|   +--rw vpn* [instance-id]
|     +--rw instance-id      lisp-at:instance-id-type
|     +--rw iid-name
|       -> /ni:network-instances/network-instance/name

```

2.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp@2021-02-22.yang"
module ietf-lisp {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp";

  prefix lisp;

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }
  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note
  import ietf-lisp-address-types {
    prefix lisp-at;
    reference "RFC XXXX: LISP YANG model";
  }
}
```

```
}
import ietf-yang-types {
  prefix yang;
  reference "RFC 6991: Common YANG Data Types";
}
import ietf-routing {
  prefix "rt";
  reference
    "RFC 8349: A YANG Data Model for Routing Management
    (NMDA version)";
}
import ietf-network-instance {
  prefix "ni";
  reference
    "RFC 8529: YANG Model for Network Instances";
}

organization
  "IETF LISP (Locator/ID Separation Protocol) Working Group";
contact
  "WG Web:  <http://tools.ietf.org/wg/lisp/>
  WG List:  <mailto:lisp@ietf.org>

  Editor:    Vina Ermagan
             <mailto:ermagan@gmail.com>

  Editor:    Alberto Rodriguez-Natal
             <mailto:natal@cisco.com>

  Editor:    Reshad Rahman
             <mailto:reshad@yahoo.com>";

description
  "This YANG module defines the generic parameters for LISP.
  The module can be extended by vendors to define vendor-specific
  LISP parameters and policies.

  Copyright (c) 2018 IETF Trust and the persons identified as
  authors of the code.  All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC XXXX; see
  the RFC itself for full legal notices.
```

```
    ";

    reference "RFC XXXX";

    revision 2021-02-22 {
      description
        "Initial revision.";
      reference
        "https://tools.ietf.org/html/rfc6830";
    }

    /*
     * Identity definitions
     */
    identity lisp {
      base "rt:control-plane-protocol";
      description "LISP protocol.";
      reference
        "RFC 6830: The Locator/ID Separation Protocol (LISP).";
    }

    identity lisp-role {
      description
        "LISP router role.";
    }
    identity itr {
      base lisp-role;
      description
        "LISP ITR.";
    }
    identity pitr {
      base lisp-role;
      description
        "LISP PITR.";
    }
    identity etr {
      base lisp-role;
      description
        "LISP ETR.";
    }
    identity petr {
      base lisp-role;
      description
        "LISP PETR.";
    }

    identity mapping-system {
      description
```

```
    "Mapping System interface";
  }
  identity single-node-mapping-system {
    base mapping-system;
    description
      "logically singular Map Server";
  }

  identity map-reply-act {
    description
      "Defines the lisp map-cache ACT type";
    reference
      "https://www.iana.org/assignments/lisp-parameters"
      + "/lisp-parameters.xhtml#lisp-act-value";
  }
  identity no-action {
    base map-reply-act;
    description
      "Mapping is kept alive and no encapsulation
      occurs.";
  }
  identity natively-forward {
    base map-reply-act;
    description
      "Matching packets are not encapsulated or
      dropped but natively forwarded.";
  }
  identity send-map-request {
    base map-reply-act;
    description
      "Matching packets invoke Map-Requests.";
  }
  identity drop-no-reason {
    base map-reply-act;
    description
      "Matching packets are dropped.";
  }
  identity drop-policy-denied {
    base map-reply-act;
    description
      "Matching packets are dropped (due to policy).";
  }
  identity drop-auth-failure {
    base map-reply-act;
    description
      "Matching packets are dropped (due to authentication
      failure).";
  }
}
```

```
identity auth-algorithm {
  description
    "Base identity for the authentication mechanisms supported by
    LISP.";
  reference
    "https://www.iana.org/assignments/lisp-parameters"
    + "/lisp-parameters.xhtml#lisp-key-id-numbers";
}
identity no-auth-algorithm {
  base auth-algorithm;
  description
    "No authentication.";
}
identity hmac-sha-1-96-none {
  base auth-algorithm;
  description
    "MAC = HMAC-SHA-1-96 (RFC2404), KDF = none";
}
identity hmac-sha-256-128-none {
  base auth-algorithm;
  description
    "MAC = HMAC-SHA-256-128 (RFC4868), KDF = none";
}
identity hmac-sha-256-128-HKDF-SHA2562 {
  base auth-algorithm;
  description
    "MAC = HMAC-SHA-256-128, KDF = HKDF-SHA2562 (RFC4868)";
}

typedef mapping-system-ref {
  type identityref {
    base mapping-system;
  }
  description
    "Mapping System reference";
}

typedef lisp-role-ref {
  type identityref {
    base lisp-role;
  }
  description
    "LISP role reference";
}

typedef map-reply-action {
  type identityref {
    base map-reply-act;
  }
}
```

```
    description
      "Map-Reply action reference";
  }
  typedef eid-id {
    type string {
      pattern '[a-zA-Z0-9\-\_\.]*';
    }
    description
      "Type encoding of lisp-addresses to be generally used in EID
      keyed lists.";
  }
  typedef auth-algorithm-type {
    type identityref {
      base auth-algorithm;
    }
    description
      "Authentication algorithm reference";
  }
  typedef xtr-id-type {
    type binary {
      length "16";
    }
    description
      "128-bit xTR identifier.";
  }

  grouping locator-properties {
    description
      "Properties of a RLOC";
    leaf priority {
      type uint8;
      description
        "Locator priority.";
    }
    leaf weight {
      type uint8;
      description
        "Locator weight.";
    }
    leaf multicast-priority {
      type uint8;
      description
        "Locator's multicast priority";
    }
    leaf multicast-weight {
      type uint8;
      description
        "Locator's multicast weight";
    }
  }
```

```
    }
  }

  grouping locators-grouping {
    description
      "Grouping that defines a list of LISP locators.";
    list locator {
      key "locator-id";
      description
        "List of routing locators";
      leaf locator-id {
        type string {
          length "1..64";
          pattern '[a-zA-Z0-9\-\_\.]*';
        }
        description
          "Locator id";
      }
      container locator-address {
        uses lisp-at:lisp-address;
        description
          "The locator address provided in LISP canonincaal
          address format.";
      }
      uses locator-properties;
    }
  }

  grouping local-locators-grouping {
    description
      "Grouping that defines a list of LISP locators.";
    list interface {
      key "interface-ref";
      description
        "The address type of the locator";
      leaf interface-ref {
        type if:interface-ref;
        description
          "The name of the interface supporting the locator.";
      }
      uses locator-properties;
    }
  }

  grouping mapping {
    description
      "Grouping that defines a LISP mapping.";
```

```
container eid {
  uses lisp-at:lisp-address;
  description
    "End-host Identifier (EID) to be mapped to a list of
    locators";
}
leaf time-to-live {
  type uint32;
  units minutes;
  description
    "Mapping validity period in minutes (as per RF6830).";
}
leaf creation-time {
  type yang:date-and-time;
  config false;
  description
    "Time when the mapping was created.";
}
leaf authoritative {
  type bits {
    bit A {
      description
        "Authoritative bit.";
    }
  }
  description
    "Bit that indicates if mapping comes from an
    authoritative source.";
}
leaf static {
  type boolean;
  default "false";
  description
    "This leaf should be true if the mapping is static.";
}
choice locator-list {
  description
    "list of locartors are either negative, or positive.";
  case negative-mapping {
    leaf map-reply-action {
      type map-reply-action;
      description
        "Forwarding action for a negative mapping.";
    }
  }
  case positive-mapping {
    container rlocs {
      uses locators-grouping;
    }
  }
}
```



```
        description
            "List of locators for a positive mapping.";
    }
}
}

grouping mappings {
    description
        "Grouping that defines a list of LISP mappings.";
    list vpn {
        key "instance-id";
        description
            "VPN to which the mappings belong.";
        leaf instance-id {
            type leafref {
                path "/rt:routing/rt:control-plane-protocols"
                    + "/rt:control-plane-protocol/lisp:lisp"
                    + "/lisp:vpns/lisp:vpn"
                    + "/lisp:instance-id";
            }
            description
                "VPN identifier.";
        }
        container mappings {
            description
                "Mappings within the VPN.";
            list mapping {
                key "eid-id";
                description
                    "List of EID to RLOCs mappings.";
                leaf eid-id {
                    type eid-id;
                    description
                        "Id that uniquely identifies a mapping.";
                }
                uses mapping;
            }
        }
    }
}

grouping auth-key {
    description "Grouping that defines authentication keys.";
    container authentication-keys {
        description "Multiple authentication keys can be defined.";
        list authentication-key {
            key "auth-key-id";
```

```
    description
    "Authentication key parameters.";
    leaf auth-key-id {
      type string {
        pattern '[a-zA-Z0-9\-\_\.]*';
      }
      description
      "Identifier of the authentication key.";
    }
    leaf-list auth-algorithm-id {
      type lisp:auth-algorithm-type;
      description
      "Authentication algorithm used with the key.";
    }
    leaf auth-key-value {
      type string;
      description
      "Clear text authentication key.";
    }
  }
}

augment "/rt:routing/rt:control-plane-protocols"
+ "/rt:control-plane-protocol" {
  when "derived-from-or-self(rt:type, 'lisp:lisp')" {
    description
    "This augmentation is only valid for a control-plane protocol
    instance of LISP.";
  }
  description "LISP protocol ietf-routing module
  control-plane-protocol augmentation.";

  container lisp {
    description
    "Parameters for the LISP subsystem.";

    container locator-sets {
      description
      "Container that defines a named locator set which can be
      referenced elsewhere.";
      list locator-set {
        key "locator-set-name";
        description
        "Multiple locator sets can be defined.";
        leaf locator-set-name {
          type string {
            length "1..64";
          }
        }
      }
    }
  }
}
```

```
        pattern '[a-zA-Z0-9\-\_\.]*';
    }
    description
        "Locator set name";
}
choice locator-type {
    description
        "Locator sets can be based on local interfaces, or
        general locators.";
    case local-interface {
        uses local-locators-grouping;
        description
            "List of locators in this set based on local
            interfaces.";
    }
    case general-locator {
        uses locators-grouping;
        description
            "List of locators in this set based on
            lisp-address.";
    }
}
}
}

list lisp-role {
    key lisp-role-type;
    description
        "List of lisp device roles such as MS, MR, ITR,
        PITR, ETR or PETR.";
    leaf lisp-role-type {
        type lisp-role-ref;
        description
            "The type of LISP device - identity derived from the
            'lisp-device' base identity.";
    }
}

container lisp-router-id {
    when "../lisp-role/lisp-role-type = 'lisp:itr' or
        ../lisp-role/lisp-role-type = 'lisp:pitr' or
        ../lisp-role/lisp-role-type = 'lisp:etr' or
        ../lisp-role/lisp-role-type = 'lisp:petr'" {
        description "Only when ITR, PITR, ETR or PETR.";
    }
    description
        "Site-ID and xTR-ID of the device.";
    leaf site-id {
```

```

    type uint64;
    description "Site ID";
}
leaf xtr-id {
    type lisp:xtr-id-type;
    description "xTR ID";
}
}

container vpns {
    when "../lisp-role/lisp-role-type = 'lisp:itr' or
        ../lisp-role/lisp-role-type = 'lisp:pitr' or
        ../lisp-role/lisp-role-type = 'lisp:etr' or
        ../lisp-role/lisp-role-type = 'lisp:petr'" {
        description "Only when ITR, PITR, ETR or PETR.";
    }
    description "VPNs";
    list vpn {
        key instance-id;
        unique "iid-name";
        description "List of VPNs";

        leaf instance-id {
            type lisp-at:instance-id-type;
            description
                "VPN identifier. The value 0 for instance-id must be
                 used for the default VRF.";
        }
        leaf iid-name {
            type leafref {
                path "/ni:network-instances/ni:network-instance"
                    + "/ni:name";
            }
            mandatory true;
            description
                "Name of VPN (e.g. VRF) to which an instance-id is
                 bound. Each instance-id is bound to a different VPN";
        }
    }
}
}
}

<CODE ENDS>
```

3. LISP-ITR Module

This module captures the configuration data model of a LISP ITR. The model also captures some operational data elements.

3.1. Module Structure

```

module: ietf-lisp-itr
augment /rt:routing/rt:control-plane-protocols
  /rt:control-plane-protocol/lisp:lisp:
    +--rw itr!
      +--rw rloc-probing!
        |   +--rw interval?          uint16
        |   +--rw retries?           uint8
        |   +--rw retries-interval?  uint16
      +--rw itr-rlocs?      leafref
      +--rw map-resolvers
        |   +--rw map-resolver*      inet:ip-address
      +--rw proxy-etrs
        |   +--rw proxy-etr-address*  inet:ip-address
      +--rw map-cache
        +--ro size?      uint32
        +--ro limit?     uint32
        +--rw vpn* [instance-id]
          +--rw instance-id      leafref
          +--rw mappings
            +--rw mapping* [eid-id]
              +--rw eid-id          eid-id
              +--rw eid
                +--rw address-type
                |   +--rw lisp-address-family-ref
                +--rw (address)?
                  +--:(no-address)
                  |   +--rw no-address?          empty
                  +--:(ipv4)
                  |   +--rw ipv4?
                  |       inet:ipv4-address
                  +--:(ipv4-prefix)
                  |   +--rw ipv4-prefix?
                  |       inet:ipv4-prefix
                  +--:(ipv6)
                  |   +--rw ipv6?
                  |       inet:ipv6-address
                  +--:(ipv6-prefix)
                  |   +--rw ipv6-prefix?
                  |       inet:ipv6-prefix
                  +--:(mac)
                  |   +--rw mac?

```

```

|               yang:mac-address
+---:(distinguished-name)
|   +---rw distinguished-name?
|       distinguished-name-type
+---:(as-number)
|   +---rw as-number?
|       inet:as-number
+---:(null-address)
|   +---rw null-address
|       +---rw address?    empty
+---:(afi-list)
|   +---rw afi-list
|       +---rw address-list*    simple-address
+---:(instance-id)
|   +---rw instance-id
|       +---rw instance-id?    instance-id-type
|       +---rw mask-length?    uint8
|       +---rw address?        simple-address
+---:(as-number-lcaf)
|   +---rw as-number-lcaf
|       +---rw as?            inet:as-number
|       +---rw address?        simple-address
+---:(application-data)
|   +---rw application-data
|       +---rw address?
|           simple-address
|       +---rw protocol?        uint8
|       +---rw ip-tos?          int32
|       +---rw local-port-low?
|           inet:port-number
|       +---rw local-port-high?
|           inet:port-number
|       +---rw remote-port-low?
|           inet:port-number
|       +---rw remote-port-high?
|           inet:port-number
+---:(geo-coordinates)
|   +---rw geo-coordinates
|       +---rw latitude?        bits
|       +---rw latitude-degrees? uint8
|       +---rw latitude-minutes? uint8
|       +---rw latitude-seconds? uint8
|       +---rw longitude?       bits
|       +---rw longitude-degrees? uint16
|       +---rw longitude-minutes? uint8
|       +---rw longitude-seconds? uint8
|       +---rw altitude?        int32
|       +---rw address?

```

```

|                                     simple-address
+---:(nat-traversal)
|   +---rw nat-traversal
|   |   +---rw ms-udp-port?          uint16
|   |   +---rw etr-udp-port?        uint16
|   |   +---rw global-etr-rloc?
|   |   |       simple-address
|   |   +---rw ms-rloc?
|   |   |       simple-address
|   |   +---rw private-etr-rloc?
|   |   |       simple-address
|   |   +---rw rtr-rlocs*
|   |       simple-address
+---:(explicit-locator-path)
|   +---rw explicit-locator-path
|   |   +---rw hop* [hop-id]
|   |   |       +---rw hop-id        string
|   |   |       +---rw address?      simple-address
|   |   |       +---rw lrs-bits?     bits
+---:(source-dest-key)
|   +---rw source-dest-key
|   |   +---rw source?      simple-address
|   |   +---rw dest?       simple-address
+---:(key-value-address)
|   +---rw key-value-address
|   |   +---rw key?        simple-address
|   |   +---rw value?     simple-address
+---:(service-path)
|   +---rw service-path
|   |   +---rw service-path-id?
|   |   |       service-path-id-type
|   |   +---rw service-index?    uint8
+---rw time-to-live?          uint32
+---ro creation-time?         yang:date-and-time
+---rw authoritative?        bits
+---rw static?                boolean
+---rw (locator-list)?
|   +---:(negative-mapping)
|   |   +---rw map-reply-action?  map-reply-action
+---:(positive-mapping)
|   +---rw rlocs
|   |   +---rw locator* [locator-id]
|   |   |       +---rw locator-id    string
|   |   |       +---rw locator-address
|   |   |       |       +---rw address-type
|   |   |       |       |       lisp-address-family-ref
|   |   |       +---rw (address)?
|   |   |       |       +---:(no-address)

```

```

    +--rw no-address?
        empty
+--:(ipv4)
    +--rw ipv4?
        inet:ipv4-address
+--:(ipv4-prefix)
    +--rw ipv4-prefix?
        inet:ipv4-prefix
+--:(ipv6)
    +--rw ipv6?
        inet:ipv6-address
+--:(ipv6-prefix)
    +--rw ipv6-prefix?
        inet:ipv6-prefix
+--:(mac)
    +--rw mac?
        yang:mac-address
+--:(distinguished-name)
    +--rw distinguished-name?
        distinguished-name-type
+--:(as-number)
    +--rw as-number?
        inet:as-number
+--:(null-address)
    +--rw null-address
    +--rw address?    empty
+--:(afi-list)
    +--rw afi-list
    +--rw address-list*
        simple-address
+--:(instance-id)
    +--rw instance-id
    +--rw instance-id?
        | instance-id-type
    +--rw mask-length?    uint8
    +--rw address?
        simple-address
+--:(as-number-lcaf)
    +--rw as-number-lcaf
    +--rw as?
        | inet:as-number
    +--rw address?
        simple-address
+--:(application-data)
    +--rw application-data
    +--rw address?
        | simple-address
    +--rw protocol?

```



```

|         uint8
+--rw ip-tos?
|         int32
+--rw local-port-low?
|         inet:port-number
+--rw local-port-high?
|         inet:port-number
+--rw remote-port-low?
|         inet:port-number
+--rw remote-port-high?
|         inet:port-number
+---:(geo-coordinates)
+--rw geo-coordinates
+--rw latitude?
|         bits
+--rw latitude-degrees?
|         uint8
+--rw latitude-minutes?
|         uint8
+--rw latitude-seconds?
|         uint8
+--rw longitude?
|         bits
+--rw longitude-degrees?
|         uint16
+--rw longitude-minutes?
|         uint8
+--rw longitude-seconds?
|         uint8
+--rw altitude?
|         int32
+--rw address?
|         simple-address
+---:(nat-traversal)
+--rw nat-traversal
+--rw ms-udp-port?
|         uint16
+--rw etr-udp-port?
|         uint16
+--rw global-etr-rloc?
|         simple-address
+--rw ms-rloc?
|         simple-address
+--rw private-etr-rloc?
|         simple-address
+--rw rtr-rlocs*
|         simple-address
+---:(explicit-locator-path)

```

```

+--rw explicit-locator-path
+--rw hop* [hop-id]
+--rw hop-id
|   string
+--rw address?
|   simple-address
+--rw lrs-bits?   bits
+--:(source-dest-key)
+--rw source-dest-key
+--rw source?
|   simple-address
+--rw dest?
|   simple-address
+--:(key-value-address)
+--rw key-value-address
+--rw key?
|   simple-address
+--rw value?
|   simple-address
+--:(service-path)
+--rw service-path
+--rw service-path-id?
|   service-path-id-type
+--rw service-index?
|   uint8
+--rw priority?           uint8
+--rw weight?             uint8
+--rw multicast-priority? uint8
+--rw multicast-weight?   uint8

```

3.2. Module Definition

```

<CODE BEGINS> file "ietf-lisp-itr@2019-02-23.yang"
module ietf-lisp-itr {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-itr";

  prefix lisp-itr;

  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note
  import ietf-lisp {
    prefix lisp;
    reference "RFC XXXX: LISP YANG model";
  }
  import ietf-inet-types {
    prefix inet;

```

```
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management
      (NMDA version)";
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/lisp/>
    WG List:  <mailto:lisp@ietf.org>

    Editor:   Vina Ermagan
              <mailto:ermagan@gmail.com>

    Editor:   Alberto Rodriguez-Natal
              <mailto:natal@cisco.com>

    Editor:   Reshad Rahman
              <mailto:reshad@yahoo.com>";
  description
    "This YANG module defines the generic parameters for a LISP
    ITR. The module can be extended by vendors to define
    vendor-specific parameters and policies.

    Copyright (c) 2018 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD License
    set forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (http://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX; see
    the RFC itself for full legal notices.
    ";

  reference "RFC XXXX";

  revision 2019-02-23 {
    description
      "Initial revision.";
    reference
```

```
    "https://tools.ietf.org/html/rfc6830";
  }
  augment "/rt:routing/rt:control-plane-protocols"
    + "/rt:control-plane-protocol/lisp:lisp" {
    when "lisp:lisp-role/lisp:lisp-role-type = 'lisp:itr' or
        lisp:lisp-role/lisp:lisp-role-type = 'lisp:pitr'" {
      description
        "Augment is valid when LISP role type is ITR or PITR.";
    }
    description
      "This augments the LISP devices list with (P)ITR specific
      parameters.";
    container itr {
      presence "LISP (P)ITR operation enabled";
      description
        "ITR parameters";
      container rloc-probing {
        presence "RLOC probing active";
        description
          "RLOC-probing parameters";
        leaf interval {
          type uint16;
          units "seconds";
          description
            "Interval in seconds for resending the probes";
        }
        leaf retries {
          type uint8;
          description
            "Number of retries for sending the probes";
        }
        leaf retries-interval {
          type uint16;
          units "seconds";
          description
            "Interval in seconds between retries when sending probes.
            The action taken if all retries fail to receive is
            implementation specific.";
        }
      }
    }
    leaf itr-rlocs {
      type leafref {
        path "/rt:routing/rt:control-plane-protocols"
          + "/rt:control-plane-protocol/lisp:lisp"
          + "/lisp:locator-sets/lisp:locator-set"
          + "/lisp:locator-set-name";
      }
      description

```

```
        "Reference to a locator set that the (P)ITR includes in
        Map-Requests";
    }
    container map-resolvers {
        description
            "Map-Resolvers that the (P)ITR uses.";
        leaf-list map-resolver {
            type inet:ip-address;
            description
                "Each Map-Resolver within the list of Map-Resolvers.";
        }
    }
    container proxy-etr {
        when "../lisp:lisp-role/lisp:lisp-role-type='lisp:itr'" {
            description
                "Container exists only when LISP role type is ITR";
        }
        description
            "Proxy ETRs that the ITR uses.";
        leaf-list proxy-etr-address {
            type inet:ip-address;
            description
                "Proxy ETR RLOC address.";
        }
    }
    container map-cache {
        leaf size {
            type uint32;
            config false;
            description
                "Current number of entries in the EID-to-RLOC map-cache";
        }
        leaf limit {
            type uint32;
            config false;
            description
                "Maximum permissible number of entries in the EID-to-RLOC
                map-cache";
        }
    }

    uses lisp:mappings;
    description
        "EID to RLOCs mappings cache.";
}
}
}
}
<CODE ENDS>
```

4. LISP-ETR Module

This module captures the configuration data model of a LISP ETR. The model also captures some operational data elements.

4.1. Module Structure

```

module: ietf-lisp-etr
augment /rt:routing/rt:control-plane-protocols
  /rt:control-plane-protocol/lisp:lisp:
  +--rw etr!
    +--rw map-servers
      +--rw map-server* [ms-address]
        +--rw ms-address          inet:ip-address
        +--rw authentication-keys
          +--rw authentication-key* [auth-key-id]
            +--rw auth-key-id      string
            +--rw auth-algorithm-id*
              | lisp:auth-algorithm-type
            +--rw auth-key-value?  string
    +--rw local-eids
      +--rw vpn* [instance-id]
      +--rw instance-id    leafref
      +--rw eids
        +--rw local-eid* [eid-id]
          +--rw eid-id          lisp:eid-id
          +--rw eid-address
            +--rw address-type
              | lisp-address-family-ref
            +--rw (address)?
              +--:(no-address)
                | +--rw no-address?          empty
              +--:(ipv4)
                | +--rw ipv4?
                  | inet:ipv4-address
              +--:(ipv4-prefix)
                | +--rw ipv4-prefix?
                  | inet:ipv4-prefix
              +--:(ipv6)
                | +--rw ipv6?
                  | inet:ipv6-address
              +--:(ipv6-prefix)
                | +--rw ipv6-prefix?
                  | inet:ipv6-prefix
              +--:(mac)
                | +--rw mac?
                  | yang:mac-address
              +--:(distinguished-name)

```

```

|      +---rw distinguished-name?
|          distinguished-name-type
+---: (as-number)
|      +---rw as-number?
|          inet:as-number
+---: (null-address)
|      +---rw null-address
|          +---rw address?    empty
+---: (afi-list)
|      +---rw afi-list
|          +---rw address-list*    simple-address
+---: (instance-id)
|      +---rw instance-id
|          +---rw instance-id?    instance-id-type
|          +---rw mask-length?    uint8
|          +---rw address?        simple-address
+---: (as-number-lcaf)
|      +---rw as-number-lcaf
|          +---rw as?            inet:as-number
|          +---rw address?        simple-address
+---: (application-data)
|      +---rw application-data
|          +---rw address?
|              simple-address
|          +---rw protocol?        uint8
|          +---rw ip-tos?          int32
|          +---rw local-port-low?
|              inet:port-number
|          +---rw local-port-high?
|              inet:port-number
|          +---rw remote-port-low?
|              inet:port-number
|          +---rw remote-port-high?
|              inet:port-number
+---: (geo-coordinates)
|      +---rw geo-coordinates
|          +---rw latitude?        bits
|          +---rw latitude-degrees? uint8
|          +---rw latitude-minutes? uint8
|          +---rw latitude-seconds? uint8
|          +---rw longitude?       bits
|          +---rw longitude-degrees? uint16
|          +---rw longitude-minutes? uint8
|          +---rw longitude-seconds? uint8
|          +---rw altitude?        int32
|          +---rw address?
|              simple-address
+---: (nat-traversal)

```

```

+--rw nat-traversal
+--rw ms-udp-port?      uint16
+--rw etr-udp-port?     uint16
+--rw global-etr-rloc?
|   simple-address
+--rw ms-rloc?
|   simple-address
+--rw private-etr-rloc?
|   simple-address
+--rw rtr-rlocs*
|   simple-address
+--:(explicit-locator-path)
+--rw explicit-locator-path
+--rw hop* [hop-id]
|   +--rw hop-id      string
|   +--rw address?    simple-address
|   +--rw lrs-bits?   bits
+--:(source-dest-key)
+--rw source-dest-key
+--rw source?          simple-address
+--rw dest?             simple-address
+--:(key-value-address)
+--rw key-value-address
+--rw key?              simple-address
+--rw value?            simple-address
+--:(service-path)
+--rw service-path
+--rw service-path-id?
|   service-path-id-type
+--rw service-index?    uint8
+--rw rlocs?             leafref
+--rw record-ttl?        uint32
+--rw want-map-notify?   boolean
+--rw proxy-reply?       boolean
+--rw registration-interval? uint16

```

4.2. Module Definition

```

<CODE BEGINS> file "ietf-lisp-etr@2021-02-22.yang"
module ietf-lisp-etr {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-etr";

  prefix lisp-etr;

  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note

```



```
import ietf-lisp {
  prefix lisp;
  reference "RFC XXXX: LISP YANG model";
}
import ietf-lisp-address-types {
  prefix lisp-at;
  reference "RFC XXXX: LISP YANG model";
}
import ietf-inet-types {
  prefix inet;
  reference "RFC 6991: Common YANG Data Types";
}
import ietf-routing {
  prefix "rt";
  reference
    "RFC 8349: A YANG Data Model for Routing Management
    (NMDA version)";
}

organization
  "IETF LISP (Locator/ID Separation Protocol) Working Group";
contact
  "WG Web:  <http://tools.ietf.org/wg/lisp/>
  WG List:  <mailto:lisp@ietf.org>

  Editor:    Vina Ermagan
             <mailto:ermagan@gmail.com>

  Editor:    Alberto Rodriguez-Natal
             <mailto:natal@cisco.com>

  Editor:    Reshad Rahman
             <mailto:reshad@yahoo.com>;

description
  "This YANG module defines the generic parameters for a LISP
  ETR. The module can be extended by vendors to define
  vendor-specific parameters and policies.

  Copyright (c) 2018 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).
```

```
    This version of this YANG module is part of RFC XXXX; see
    the RFC itself for full legal notices.
";

reference "RFC XXXX";

revision 2021-02-22 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6830";
}
augment "/rt:routing/rt:control-plane-protocols"
  + "/rt:control-plane-protocol/lisp:lisp" {
  when "lisp:lisp-role/lisp:lisp-role-type = 'lisp:etr' or
    lisp:lisp-role/lisp:lisp-role-type = 'lisp:petr'" {
    description
      "Augment is valid when LISP device type is (P)ETR.";
  }
  description
    "This augments the LISP devices list with (P)ETR specific
    parameters.";
  container etr {
    presence "LISP (P)ETR operation enabled";
    description
      "(P)ETR parameters.";

    container map-servers {
      when "../lisp:lisp-role/lisp:lisp-role-type='lisp:etr'" {
        description
          "Container exists only when LISP device type is ETR.";
      }
      description
        "Map-Servers that the ETR uses.";
      list map-server {
        key "ms-address";
        description
          "Each Map-Server within the list of Map-Servers.";
        leaf ms-address {
          type inet:ip-address;
          description
            "Map-Server address.";
        }
        uses lisp:auth-key;
      }
    }
  }

  container local-eids {
```

```
when "../../../lisp:lisp-role/lisp:lisp-role-type='lisp:etr'" {
  description
    "Container exists only when LISP device type is ETR.";
}
description
  "VPNs served by the ETR.";
list vpn {
  key "instance-id";
  description
    "VPN for local-EIDs.";
  leaf instance-id {
    type leafref {
      path "/rt:routing/rt:control-plane-protocols"
        + "/rt:control-plane-protocol/lisp:lisp"
        + "/lisp:vpns/lisp:vpn"
        + "/lisp:instance-id";
    }
    description
      "VPN identifier.";
  }
  container eids {
    description
      "EIDs served by the ETR.";
    list local-eid {
      key "eid-id";
      description
        "List of local EIDs.";
      leaf eid-id {
        type lisp:eid-id;
        description
          "Unique id of local EID.";
      }
      container eid-address {
        uses lisp-at:lisp-address;
        description
          "EID address in generic LISP address format.";
      }
      leaf rlocs {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols"
            + "/rt:control-plane-protocol/lisp:lisp"
            + "/lisp:locator-sets/lisp:locator-set"
            + "/lisp:locator-set-name";
        }
        description
          "Locator set mapped to this local EID.";
      }
      leaf record-ttl {
```

```

        type uint32;
        units minutes;
        description
            "Validity period of the EID to RLOCs mapping
             provided in Map-Replies.";
    }
    leaf want-map-notify {
        type boolean;
        default "true";
        description
            "Flag which if set in a Map-Register requests that
             a Map-Notify be sent in response.";
    }
    leaf proxy-reply {
        type boolean;
        default "false";
        description
            "Flag which if set in a Map-Register requests that
             the Map-Server proxy Map-Replies for the ETR.";
    }
    leaf registration-interval {
        type uint16;
        units "seconds";
        default "60";
        description
            "Interval between consecutive Map-Registers.";
    }
}
}
}
}
}
}
}
<CODE ENDS>
```

5. LISP-Map-Server Module

This module captures the configuration data model of a LISP Map Server [RFC6833]. The model also captures some operational data elements.

5.1. Module Structure

```

module: ietf-lisp-mapserver
augment /rt:routing/rt:control-plane-protocols
  /rt:control-plane-protocol/lisp:lisp:
    +--rw map-server!
      +--rw sites
        +--rw site* [site-id]
          +--rw site-id          uint64
          +--rw authentication-keys
            +--rw authentication-key* [auth-key-id]
              +--rw auth-key-id      string
              +--rw auth-algorithm-id*
                | lisp:auth-algorithm-type
              +--rw auth-key-value?   string
          +--rw xtr-ids* [xtr-id]
            +--rw xtr-id            uint64
            +--rw authentication-keys
              +--rw authentication-key* [auth-key-id]
                +--rw auth-key-id      string
                +--rw auth-algorithm-id*
                  | lisp:auth-algorithm-type
                +--rw auth-key-value?   string
      +--rw vpns
        +--rw vpn* [instance-id]
          +--rw instance-id      lisp-at:instance-id-type
          +--rw mappings
            +--rw mapping* [eid-id]
              +--rw eid-id          lisp:eid-id
              +--rw eid-address
                +--rw address-type
                  | lisp-address-family-ref
                +--rw (address)?
                  +--:(no-address)
                  | +--rw no-address?          empty
                  +--:(ipv4)
                  | +--rw ipv4?
                  |   inet:ipv4-address
                  +--:(ipv4-prefix)
                  | +--rw ipv4-prefix?
                  |   inet:ipv4-prefix
                  +--:(ipv6)
                  | +--rw ipv6?
                  |   inet:ipv6-address
                  +--:(ipv6-prefix)
                  | +--rw ipv6-prefix?
                  |   inet:ipv6-prefix
                  +--:(mac)
                  | +--rw mac?
                  |   yang:mac-address

```

```

+---:(distinguished-name)
|   +---rw distinguished-name?
|       distinguished-name-type
+---:(as-number)
|   +---rw as-number?
|       inet:as-number
+---:(null-address)
|   +---rw null-address
|       +---rw address?    empty
+---:(afi-list)
|   +---rw afi-list
|       +---rw address-list*    simple-address
+---:(instance-id)
|   +---rw instance-id
|       +---rw instance-id?    instance-id-type
|       +---rw mask-length?    uint8
|       +---rw address?        simple-address
+---:(as-number-lcaf)
|   +---rw as-number-lcaf
|       +---rw as?            inet:as-number
|       +---rw address?        simple-address
+---:(application-data)
|   +---rw application-data
|       +---rw address?
|           |
|           |   simple-address
|       +---rw protocol?        uint8
|       +---rw ip-tos?          int32
|       +---rw local-port-low?
|           |
|           |   inet:port-number
|       +---rw local-port-high?
|           |
|           |   inet:port-number
|       +---rw remote-port-low?
|           |
|           |   inet:port-number
|       +---rw remote-port-high?
|           |
|           |   inet:port-number
+---:(geo-coordinates)
|   +---rw geo-coordinates
|       +---rw latitude?        bits
|       +---rw latitude-degrees? uint8
|       +---rw latitude-minutes? uint8
|       +---rw latitude-seconds? uint8
|       +---rw longitude?       bits
|       +---rw longitude-degrees? uint16
|       +---rw longitude-minutes? uint8
|       +---rw longitude-seconds? uint8
|       +---rw altitude?        int32
|       +---rw address?
|           |
|           |   simple-address

```

```

+---:(nat-traversal)
|   +---rw nat-traversal
|       +---rw ms-udp-port?          uint16
|       +---rw etr-udp-port?        uint16
|       +---rw global-etr-rloc?
|           |
|           |   simple-address
|       +---rw ms-rloc?
|           |
|           |   simple-address
|       +---rw private-etr-rloc?
|           |
|           |   simple-address
|       +---rw rtr-rlocs*
|           |
|           |   simple-address
+---:(explicit-locator-path)
|   +---rw explicit-locator-path
|       +---rw hop* [hop-id]
|           +---rw hop-id          string
|           +---rw address?        simple-address
|           +---rw lrs-bits?       bits
+---:(source-dest-key)
|   +---rw source-dest-key
|       +---rw source?            simple-address
|       +---rw dest?             simple-address
+---:(key-value-address)
|   +---rw key-value-address
|       +---rw key?              simple-address
|       +---rw value?            simple-address
+---:(service-path)
|   +---rw service-path
|       +---rw service-path-id?
|           |
|           |   service-path-id-type
|       +---rw service-index?     uint8
+---rw site-id*                  uint64
+---rw more-specifics-accepted?  boolean
+---rw mapping-expiration-timeout? int16
+---ro first-registration-time?
|   |
|   |   yang:date-and-time
+---ro last-registration-time?
|   |
|   |   yang:date-and-time
+---rw mapping-records
|   +---rw mapping-record* [xtr-id]
|       +---rw xtr-id
|           |
|           |   lisp:xtr-id-type
|       +---rw site-id?          uint64
+---rw eid
|   +---rw address-type
|       |
|       |   lisp-address-family-ref
|       +---rw (address)?
|           |
|           |   +---:(no-address)

```

```

+---rw no-address?
    empty
+---:(ipv4)
+---rw ipv4?
    inet:ipv4-address
+---:(ipv4-prefix)
+---rw ipv4-prefix?
    inet:ipv4-prefix
+---:(ipv6)
+---rw ipv6?
    inet:ipv6-address
+---:(ipv6-prefix)
+---rw ipv6-prefix?
    inet:ipv6-prefix
+---:(mac)
+---rw mac?
    yang:mac-address
+---:(distinguished-name)
+---rw distinguished-name?
    distinguished-name-type
+---:(as-number)
+---rw as-number?
    inet:as-number
+---:(null-address)
+---rw null-address
    +---rw address?    empty
+---:(afi-list)
+---rw afi-list
    +---rw address-list*
        simple-address
+---:(instance-id)
+---rw instance-id
    +---rw instance-id?
        |
        | instance-id-type
+---rw mask-length?    uint8
+---rw address?
    simple-address
+---:(as-number-lcaf)
+---rw as-number-lcaf
    +---rw as?          inet:as-number
    +---rw address?     simple-address
+---:(application-data)
+---rw application-data
    +---rw address?
        |
        | simple-address
+---rw protocol?      uint8
+---rw ip-tos?         int32
+---rw local-port-low?

```



```

|         inet:port-number
+---rw local-port-high?
|         inet:port-number
+---rw remote-port-low?
|         inet:port-number
+---rw remote-port-high?
|         inet:port-number
+---:(geo-coordinates)
+---rw geo-coordinates
+---rw latitude? bits
+---rw latitude-degrees?
|         uint8
+---rw latitude-minutes?
|         uint8
+---rw latitude-seconds?
|         uint8
+---rw longitude? bits
+---rw longitude-degrees?
|         uint16
+---rw longitude-minutes?
|         uint8
+---rw longitude-seconds?
|         uint8
+---rw altitude?
|         int32
+---rw address?
|         simple-address
+---:(nat-traversal)
+---rw nat-traversal
+---rw ms-udp-port?
|         uint16
+---rw etr-udp-port?
|         uint16
+---rw global-etr-rloc?
|         simple-address
+---rw ms-rloc?
|         simple-address
+---rw private-etr-rloc?
|         simple-address
+---rw rtr-rlocs*
|         simple-address
+---:(explicit-locator-path)
+---rw explicit-locator-path
+---rw hop* [hop-id]
|         +---rw hop-id string
|         +---rw address?
|         |         simple-address
+---rw lrs-bits? bits

```

```

+---:(source-dest-key)
|   +---rw source-dest-key
|       +---rw source?    simple-address
|       +---rw dest?     simple-address
+---:(key-value-address)
|   +---rw key-value-address
|       +---rw key?      simple-address
|       +---rw value?   simple-address
+---:(service-path)
|   +---rw service-path
|       +---rw service-path-id?
|           service-path-id-type
|       +---rw service-index?    uint8
+---rw time-to-live?            uint32
+---ro creation-time?
|   yang:date-and-time
+---rw authoritative?          bits
+---rw static?                 boolean
+---rw (locator-list)?
+---:(negative-mapping)
|   +---rw map-reply-action?
|       map-reply-action
+---:(positive-mapping)
+---rw rlocs
|   +---rw locator* [locator-id]
|       +---rw locator-id
|           string
|   +---rw locator-address
|       +---rw address-type
|           lisp-address-family-ref
|   +---rw (address)?
|       +---:(no-address)
|           +---rw no-address?
|               empty
|       +---:(ipv4)
|           +---rw ipv4?
|               inet:ipv4-address
|       +---:(ipv4-prefix)
|           +---rw ipv4-prefix?
|               inet:ipv4-prefix
|       +---:(ipv6)
|           +---rw ipv6?
|               inet:ipv6-address
|       +---:(ipv6-prefix)
|           +---rw ipv6-prefix?
|               inet:ipv6-prefix
|       +---:(mac)
|           +---rw mac?

```

```

|           yang:mac-address
+---:(distinguished-name)
|   +---rw distinguished-name?
|       distinguished-name-type
+---:(as-number)
|   +---rw as-number?
|       inet:as-number
+---:(null-address)
|   +---rw null-address
|       +---rw address?
|           empty
+---:(afi-list)
|   +---rw afi-list
|       +---rw address-list*
|           simple-address
+---:(instance-id)
|   +---rw instance-id
|       +---rw instance-id?
|           instance-id-type
|   +---rw mask-length?
|       uint8
|   +---rw address?
|       simple-address
+---:(as-number-lcaf)
|   +---rw as-number-lcaf
|       +---rw as?
|           inet:as-number
|   +---rw address?
|       simple-address
+---:(application-data)
|   +---rw application-data
|       +---rw address?
|           simple-address
|   +---rw protocol?
|       uint8
|   +---rw ip-tos?
|       int32
|   +---rw local-port-low?
|       inet:port-number
|   +---rw local-port-high?
|       inet:port-number
|   +---rw remote-port-low?
|       inet:port-number
|   +---rw remote-port-high?
|       inet:port-number
+---:(geo-coordinates)
|   +---rw geo-coordinates
|       +---rw latitude?

```

```

|         bits
+---rw latitude-degrees?
|         uint8
+---rw latitude-minutes?
|         uint8
+---rw latitude-seconds?
|         uint8
+---rw longitude?
|         bits
+---rw longitude-degrees?
|         uint16
+---rw longitude-minutes?
|         uint8
+---rw longitude-seconds?
|         uint8
+---rw altitude?
|         int32
+---rw address?
|         simple-address
+---:(nat-traversal)
+---rw nat-traversal
+---rw ms-udp-port?
|         uint16
+---rw etr-udp-port?
|         uint16
+---rw global-etr-rloc?
|         simple-address
+---rw ms-rloc?
|         simple-address
+---rw private-etr-rloc?
|         simple-address
+---rw rtr-rlocs*
|         simple-address
+---:(explicit-locator-path)
+---rw explicit-locator-path
+---rw hop* [hop-id]
|         string
+---rw address?
|         simple-address
+---rw lrs-bits?
|         bits
+---:(source-dest-key)
+---rw source-dest-key
+---rw source?
|         simple-address
+---rw dest?
|         simple-address

```

```

+--:(key-value-address)
+--rw key-value-address
+--rw key?
|
|       simple-address
+--rw value?
|
|       simple-address
+--:(service-path)
+--rw service-path
+--rw service-path-id?
|
|       service-path-id-type
+--rw service-index?
|
|       uint8
+--rw priority?
|
|       uint8
+--rw weight?
|
|       uint8
+--rw multicast-priority?
|
|       uint8
+--rw multicast-weight?
|
|       uint8
+--ro counters
+--ro map-registers-in?                yang:counter64
+--ro map-registers-in-auth-failed?    yang:counter64
+--ro map-notify-records-out?           yang:counter64
+--ro proxy-reply-records-out?          yang:counter64
+--ro map-requests-forwarded-out?       yang:counter64
+--rw mapping-system-type?    lisp:mapping-system-ref
+--ro summary
+--ro number-configured-sites?    uint32
+--ro number-registered-sites?    uint32
+--ro af-datum
+--ro af-data* [address-type]
+--ro address-type
|
|       lisp-at:lisp-address-family-ref
+--ro number-configured-eids?    uint32
+--ro number-registered-eids?    uint32
+--ro counters
+--ro map-registers-in?                yang:counter64
+--ro map-registers-in-auth-failed?    yang:counter64
+--ro map-notify-records-out?           yang:counter64
+--ro proxy-reply-records-out?          yang:counter64
+--ro map-requests-forwarded-out?       yang:counter64

```

5.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp-mapserver@2021-02-22.yang"
module ietf-lisp-mapserver {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver";

  prefix lisp-ms;

  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note
  import ietf-lisp {
    prefix lisp;
    reference "RFC XXXX: LISP YANG model";
  }
  import ietf-lisp-address-types {
    prefix lisp-at;
    reference "RFC XXXX: LISP YANG model";
  }
  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management
       (NMDA version)";
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/lisp/>
    WG List:  <mailto:lisp@ietf.org>

    Editor:   Vina Ermagan
              <mailto:ermagan@gmail.com>

    Editor:   Alberto Rodriguez-Natal
              <mailto:natal@cisco.com>

    Editor:   Reshad Rahman
              <mailto:reshad@yahoo.com>";
  description
    "This YANG module defines the generic parameters for a LISP
    Map-Server. The module can be extended by vendors to define
    vendor-specific parameters and policies."
```

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.

```
";

reference "RFC XXXX";

revision 2021-02-22 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6833";
}

identity ms {
  base lisp:lisp-role;
  description
    "LISP Map-Server.";
}

grouping ms-counters {
  description "Grouping that defines map-server counters.";
  container counters {
    config false;
    description "Container for the counters";

    leaf map-registers-in {
      type yang:counter64;
      description "Number of incoming Map-Register messages";
    }

    leaf map-registers-in-auth-failed {
      type yang:counter64;
      description
        "Number of incoming Map-Register messages failed
        authentication";
    }

    leaf map-notify-records-out {
```

```
        type yang:counter64;
        description
            "Number of outgoing Map-Notify records";
    }

    leaf proxy-reply-records-out {
        type yang:counter64;
        description
            "Number of outgoing proxy Map-Reply records";
    }

    leaf map-requests-forwarded-out {
        type yang:counter64;
        description
            "Number of outgoing Map-Requests forwarded to ETR";
    }
}

augment "/rt:routing/rt:control-plane-protocols"
+ "/rt:control-plane-protocol/lisp:lisp" {
    when "lisp:lisp-role/lisp:lisp-role-type = 'lisp-ms:ms'" {
        description
            "Augment is valid when LISP device type is Map-Server.";
    }
    description
        "This augments the LISP devices list with Map-Server
        specific parameters.";
    container map-server {
        presence "LISP Map-Server operation enabled";
        description
            "Map-Server parameters.";
        container sites{
            description
                "Sites to accept registrations from.";
            list site {
                key site-id;
                description
                    "Site that can send registrations.";
                leaf site-id {
                    type uint64;
                    description "Site ID";
                }
            }
            uses lisp:auth-key;
            list xtr-ids {
                key xtr-id;
                description "xTR-ID specific configuration.";
                leaf xtr-id {
```



```
        type uint64;
        description "xTR ID";
    }
    uses lisp:auth-key;
}
}
}
container vpns {
    description
        "VPNs for which the Map-Server accepts registrations.";
    list vpn {
        key "instance-id";
        description
            "VPN instances in the Map-Server.";
        leaf instance-id {
            type lisp-at:instance-id-type;
            description
                "VPN identifier.";
        }
        container mappings {
            description
                "EIDs registered by device.";
            list mapping {
                key "eid-id";
                description
                    "List of EIDs registered by device.";
                leaf eid-id {
                    type lisp:eid-id;
                    description
                        "Id of the EID registered.";
                }
            }
            container eid-address {
                uses lisp-at:lisp-address;
                description
                    "EID in generic LISP address format registered
                    with the Map-Server.";
            }
            leaf-list site-id {
                type uint64;
                description "Site ID";
            }
            leaf more-specifics-accepted {
                type boolean;
                default "false";
                description
                    "Flag indicating if more specific prefixes
                    can be registered.";
            }
        }
    }
}
```

```
    leaf mapping-expiration-timeout {
      type int16;
      units "seconds";
      default "180"; //3 times the mapregister int
      description
        "Time before mapping is expired if no new
         registrations are received.";
    }
    leaf first-registration-time {
      type yang:date-and-time;
      config false;
      description
        "Time at which the first registration for this
         EID was received";
    }
    leaf last-registration-time {
      type yang:date-and-time;
      config false;
      description
        "Time at which the last registration for this EID
         was received";
    }
    container mapping-records {
      description
        "Datastore of registered mappings.";
      list mapping-record {
        key xtr-id;
        description
          "Registered mapping.";
        leaf xtr-id {
          type lisp:xtr-id-type;
          description "xTR ID";
        }
        leaf site-id {
          type uint64;
          description "Site ID";
        }
        uses lisp:mapping;
      }
    }
  }
  uses ms-counters;
}

leaf mapping-system-type {
  type lisp:mapping-system-ref;
  description
```

```
        "A reference to the mapping system";
    }

    container summary {
        config false;
        description "Summary state information";

        leaf number-configured-sites {
            type uint32;
            description "Number of configured LISP sites";
        }
        leaf number-registered-sites {
            type uint32;
            description "Number of registered LISP sites";
        }
    }
    container af-datum {
        description "Number of configured EIDs per each AF";

        list af-data {
            key "address-type";
            description "Number of configured EIDs for this AF";
            leaf address-type {
                type lisp-at:lisp-address-family-ref;
                description "AF type";
            }
            leaf number-configured-eids {
                type uint32;
                description "Number of configured EIDs for this AF";
            }
            leaf number-registered-eids {
                type uint32;
                description "Number of registered EIDs for this AF";
            }
        }
    }
    }
    }
    uses ms-counters;
}
}
}
<CODE ENDS>
```

6. LISP-Map-Resolver Module

This module captures the configuration data model of a LISP Map Resolver [RFC6833]. The model also captures some operational data elements.

6.1. Module Structure

```
module: ietf-lisp-mapresolver
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/lisp:lisp:
    +--rw map-resolver!
      +--rw mapping-system-type?    lisp:mapping-system-ref
      +--rw ms-address?             inet:ip-address
```

6.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp-mapresolver@2019-02-23.yang"
module ietf-lisp-mapresolver {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-mapresolver";

  prefix lisp-mr;

  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note
  import ietf-lisp {
    prefix lisp;
    reference "RFC XXXX: LISP YANG model";
  }
  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management
       (NMDA version)";
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/lisp/>
     WG List: <mailto:lisp@ietf.org>

     Editor:  Vina Ermagan
              <mailto:ermagan@gmail.com>

     Editor:  Alberto Rodriguez-Natal
              <mailto:natal@cisco.com>
```

```
Editor:   Reshad Rahman
         <mailto:reshad@yahoo.com>;

description
  "This YANG module defines the generic parameters for a LISP
  Map-Resolver. The module can be extended by vendors to define
  vendor-specific parameters and policies.

  Copyright (c) 2018 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC XXXX; see
  the RFC itself for full legal notices.
";

reference "RFC XXXX";

revision 2019-02-23 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6833";
}
identity mr {
  base lisp:lisp-role;
  description
    "LISP Map-Resolver.";
}

augment "/rt:routing/rt:control-plane-protocols"
  + "/rt:control-plane-protocol/lisp:lisp" {
  when "lisp:lisp-role/lisp:lisp-role-type = 'lisp-mr'" {
    description
      "Augment is valid when LISP device type is Map-Resolver.";
  }
  description
    "This augments the LISP devices list with Map-Resolver
    specific parameters.";
  container map-resolver {
    presence "LISP Map-Resolver operation enabled";
    description
      "Map-Resolver parameters.";
  }
}
```

```
    leaf mapping-system-type {
      type lisp:mapping-system-ref;
      description
        "A reference to the mapping system";
    }
    leaf ms-address {
      when "../mapping-system-type="
        + "'lisp:single-node-mapping-system'";
      type inet:ip-address;
      description
        "address to reach the Map Server when "
        + "lisp-mr:single-node-mapping-system is being used.";
    }
  }
}
}
<CODE ENDS>
```

7. LISP-Address-Types Module

This module captures the various LISP address types, and is an essential building block used in other LISP modules.

7.1. Module Definition

```
<CODE BEGINS> file "ietf-lisp-address-types@2021-02-22.yang"
module ietf-lisp-address-types {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types";

  prefix lisp-at;

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types";
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/lisp/>
    WG List:  <mailto:lisp@ietf.org>
```

Editor: Vina Ermagan
<mailto:ermagan@gmail.com>

Editor: Alberto Rodriguez-Natal
<mailto:natal@cisco.com>

Editor: Reshad Rahman
<mailto:reshad@yahoo.com>;

description

"This YANG module defines the LISP Canonical Address Formats (LCAF) for LISP. The module can be extended by vendors to define vendor-specific parameters.

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.

```
";
// RFC Ed.: replace XXXX with actual RFC number and remove
// this note
reference "RFC XXXX";

revision 2021-02-22 {
  description
    "Initial revision.";
  reference
    "RFC8060: LISP Canonical Address Format (LCAF)";
}
identity lisp-address-family {
  description
    "Base identity from which identities describing LISP address
    families are derived.";
}
identity no-address-afi {
  base lisp-address-family;
  description
    "IANA Reserved.";
}
identity ipv4-afi {
```

```
    base lisp-address-family;
    description
      "IANA IPv4 address family.";
  }
  identity ipv4-prefix-afi {
    base lisp-address-family;
    description
      "IANA IPv4 address family prefix.";
  }
  identity ipv6-afi {
    base lisp-address-family;
    description
      "IANA IPv6 address family.";
  }
  identity ipv6-prefix-afi {
    base lisp-address-family;
    description
      "IANA IPv6 address family prefix.";
  }
  identity mac-afi {
    base lisp-address-family;
    description
      "IANA MAC address family.";
  }
  identity distinguished-name-afi {
    base lisp-address-family;
    description
      "IANA Distinguished Name address family.";
  }
  identity as-number-afi {
    base lisp-address-family;
    description
      "IANA AS Number address family.";
  }
  identity lcaf {
    base lisp-address-family;
    description
      "IANA LISP Canonical Address Format address family.";
  }
  identity null-address-lcaf {
    base lcaf;
    description
      "Null body LCAF type.";
  }
  identity afi-list-lcaf {
    base lcaf;
    description
      "AFI-List LCAF type.";
```



```
    }
    identity instance-id-lcaf {
      base lcaf;
      description
        "Instance-ID LCAF type.";
    }
    identity as-number-lcaf {
      base lcaf;
      description
        "AS Number LCAF type.";
    }
    identity application-data-lcaf {
      base lcaf;
      description
        "Application Data LCAF type.";
    }
    identity geo-coordinates-lcaf {
      base lcaf;
      description
        "Geo-coordinates LCAF type.";
    }
    identity opaque-key-lcaf {
      base lcaf;
      description
        "Opaque Key LCAF type.";
    }
    identity nat-traversal-lcaf {
      base lcaf;
      description
        "NAT-Traversal LCAF type.";
    }
    identity nonce-locator-lcaf {
      base lcaf;
      description
        "Nonce-Locator LCAF type.";
    }
    identity multicast-info-lcaf {
      base lcaf;
      description
        "Multicast Info LCAF type.";
    }
    identity explicit-locator-path-lcaf {
      base lcaf;
      description
        "Explicit Locator Path LCAF type.";
    }
    identity security-key-lcaf {
      base lcaf;
```

```
    description
      "Security Key LCAF type.";
  }
  identity source-dest-key-lcaf {
    base lcaf;
    description
      "Source/Dest LCAF type.";
  }
  identity replication-list-lcaf {
    base lcaf;
    description
      "Replication-List LCAF type.";
  }
  identity json-data-model-lcaf {
    base lcaf;
    description
      "JSON Data Model LCAF type.";
  }
  identity key-value-address-lcaf {
    base lcaf;
    description
      "Key/Value Address LCAF type.";
  }
  identity encapsulation-format-lcaf {
    base lcaf;
    description
      "Encapsulation Format LCAF type.";
  }
  identity service-path-lcaf {
    base lcaf;
    description
      "Service Path LCAF type.";
  }
  typedef instance-id-type {
    type uint32 {
      range "0..16777215";
    }
    description
      "Defines the range of values for an Instance ID.";
  }
  typedef service-path-id-type {
    type uint32 {
      range "0..16777215";
    }
    description
      "Defines the range of values for a Service Path ID.";
  }
  typedef distinguished-name-type {
```

```
    type string;
    description
      "Distinguished Name address.";
    reference
      "http://www.iana.org/assignments/address-family-numbers/
      address-family-numbers.xhtml";
  }
  typedef simple-address {
    type union {
      type inet:ip-address;
      type inet:ip-prefix;
      type yang:mac-address;
      type distinguished-name-type;
      type inet:as-number;
    }
    description
      "Union of address types that can be part of LCAFs.";
  }
  typedef lisp-address-family-ref {
    type identityref {
      base lisp-address-family;
    }
    description
      "LISP address family reference.";
  }
  typedef lcaf-ref {
    type identityref {
      base lcaf;
    }
    description
      "LCAF types reference.";
  }

  grouping lisp-address {
    description
      "Generic LISP address.";
    leaf address-type {
      type lisp-address-family-ref;
      mandatory true;
      description
        "Type of the LISP address.";
    }
    choice address {
      description
        "Various LISP address types, including IP, MAC, and LCAF.";

      leaf no-address {
        when "../address-type = 'lisp-at:no-address-afi'" {
```

```
        description
            "When AFI is 0.";
    }
    type empty;
    description
        "No address.";
}
leaf ipv4 {
    when "../address-type = 'lisp-at:ipv4-afi'" {
        description
            "When AFI is IPv4.";
    }
    type inet:ipv4-address;
    description
        "IPv4 address.";
}
leaf ipv4-prefix {
    when "../address-type = 'lisp-at:ipv4-prefix-afi'" {
        description
            "When AFI is IPv4.";
    }
    type inet:ipv4-prefix;
    description
        "IPv4 prefix.";
}
leaf ipv6 {
    when "../address-type = 'lisp-at:ipv6-afi'" {
        description
            "When AFI is IPv6.";
    }
    type inet:ipv6-address;
    description
        "IPv6 address.";
}
leaf ipv6-prefix {
    when "../address-type = 'lisp-at:ipv6-prefix-afi'" {
        description
            "When AFI is IPv6.";
    }
    type inet:ipv6-prefix;
    description
        "IPv6 address.";
}
leaf mac {
    when "../address-type = 'lisp-at:mac-afi'" {
        description
            "When AFI is MAC.";
    }
}
```

```
    type yang:mac-address;
    description
      "MAC address.";
  }
  leaf distinguished-name {
    when "../address-type = 'lisp-at:distinguished-name-afi'" {
      description
        "When AFI is distinguished-name.";
    }
    type distinguished-name-type;
    description
      "Distinguished Name address.";
  }
  leaf as-number {
    when "../address-type = 'lisp-at:as-number-afi'" {
      description
        "When AFI is as-number.";
    }
    type inet:as-number;
    description
      "AS Number.";
  }
  container null-address {
    when "../address-type = 'lisp-at:null-address-lcaf'" {
      description
        "When LCAF type is null.";
    }
    description
      "Null body LCAF type";
    leaf address {
      type empty;
      description
        "AFI address.";
    }
  }
  container afi-list {
    when "../address-type = 'lisp-at:afi-list-lcaf'" {
      description
        "When LCAF type is AFI-List.";
    }
    description
      "AFI-List LCAF type.";
    reference
      "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10#section-4.16.1";
    leaf-list address-list {
      type simple-address;
      description
```

```
        "List of AFI addresses.";
    }
}
container instance-id {
    when "../address-type = 'lisp-at:instance-id-lcaf'" {
        description
            "When LCAF type is Instance ID as per RFC8060.";
    }
    description
        "Instance ID LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
        #section-4.2";
    leaf instance-id {
        type instance-id-type;
        description
            "Instance ID value.";
    }
    leaf mask-length {
        type uint8;
        description
            "Mask length.";
    }
    leaf address {
        type simple-address;
        description
            "AFI address.";
    }
}
container as-number-lcaf {
    when "../address-type = 'lisp-at:as-number-lcaf'" {
        description
            "When LCAF type is AS-Number.";
    }
    description
        "AS Number LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
        #section-4.3";
    leaf as {
        type inet:as-number;
        description
            "AS number.";
    }
    leaf address {
        type simple-address;
        description
            "AFI address.";
```

```
    }  
  }  
  container application-data {  
    when "../address-type = 'lisp-at:application-data-lcaf'" {  
      description  
        "When LCAF type is Application Data.";  
    }  
    description  
      "Application Data LCAF type.";  
    reference  
      "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10  
      #section-4.4";  
    leaf address {  
      type simple-address;  
      description  
        "AFI address.";  
    }  
    leaf protocol {  
      type uint8;  
      description  
        "Protocol number.";  
    }  
    leaf ip-tos {  
      type int32;  
      description  
        "Type of service field.";  
    }  
    leaf local-port-low {  
      type inet:port-number;  
      description  
        "Low end of local port range.";  
    }  
    leaf local-port-high {  
      type inet:port-number;  
      description  
        "High end of local port range.";  
    }  
    leaf remote-port-low {  
      type inet:port-number;  
      description  
        "Low end of remote port range.";  
    }  
    leaf remote-port-high {  
      type inet:port-number;  
      description  
        "High end of remote port range.";  
    }  
  }  
}
```

```
container geo-coordinates {
  when "../address-type = 'lisp-at:geo-coordinates-lcaf'" {
    description
      "When LCAF type is Geo-coordinates.";
  }
  description
    "Geo-coordinates LCAF type. Coordinates are specified
    using the WGS 84 (World Geodetic System 1984) reference
    coordinate system";
  reference
    "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
    #section-4.5";
  leaf latitude {
    type bits {
      bit N {
        description
          "Latitude bit.";
      }
    }
    description
      "Bit that selects between North and South latitude.";
  }
  leaf latitude-degrees {
    type uint8 {
      range "0 .. 90";
    }
    description
      "Degrees of latitude.";
  }
  leaf latitude-minutes {
    type uint8 {
      range "0..59";
    }
    description
      "Minutes of latitude.";
  }
  leaf latitude-seconds {
    type uint8 {
      range "0..59";
    }
    description
      "Seconds of latitude.";
  }
  leaf longitude {
    type bits {
      bit E {
        description
          "Longitude bit.";
      }
    }
  }
}
```



```
    }
  }
  description
    "Bit that selects between East and West longitude.";
}
leaf longitude-degrees {
  type uint16 {
    range "0 .. 180";
  }
  description
    "Degrees of longitude.";
}
leaf longitude-minutes {
  type uint8 {
    range "0..59";
  }
  description
    "Minutes of longitude.";
}
leaf longitude-seconds {
  type uint8 {
    range "0..59";
  }
  description
    "Seconds of longitude.";
}
leaf altitude {
  type int32;
  description
    "Height relative to sea level in meters.";
}
leaf address {
  type simple-address;
  description
    "AFI address.";
}
}
container nat-traversal {
  when "../address-type = 'lisp-at:nat-traversal-lcaf'" {
    description
      "When LCAF type is NAT-Traversal.";
  }
  description
    "NAT-Traversal LCAF type.";
  reference
    "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
    #section-4.6";
  leaf ms-udp-port {
```

```
        type uint16;
        description
            "Map-Server UDP port (set to 4342).";
    }
    leaf etr-udp-port {
        type uint16;
        description
            "ETR UDP port.";
    }
    leaf global-etr-rloc {
        type simple-address;
        description
            "Global ETR RLOC address.";
    }
    leaf ms-rloc {
        type simple-address;
        description
            "Map-Server RLOC address.";
    }
    leaf private-etr-rloc {
        type simple-address;
        description
            "Private ETR RLOC address.";
    }
    leaf-list rtr-rlocs {
        type simple-address;
        description
            "List of RTR RLOC addresses.";
    }
}

container explicit-locator-path {
    when "../address-type = 'lisp-at:explicit-locator-path-lcaf'" {
        description
            "When LCAF type type is Explicit Locator Path.";
    }
    description
        "Explicit Locator Path LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10#section-4.9";
    list hop {
        key "hop-id";
        ordered-by user;
        description
            "List of locator hops forming the explicit path.";
        leaf hop-id {
            type string {
                length "1..64";
            }
        }
    }
}
```

```
        pattern '[a-zA-Z0-9\-\_\.]*';
    }
    description
        "Unique identifier for the hop.";
}
leaf address {
    type simple-address;
    description
        "AFI address.";
}
leaf lrs-bits {
    type bits{
        bit lookup {
            description
                "Lookup bit.";
        }
        bit rloc-probe {
            description
                "RLOC-probe bit.";
        }
        bit strict {
            description
                "Strict bit.";
        }
    }
    description
        "Flag bits per hop.";
}
}
}
container source-dest-key {
    when "../address-type = 'lisp-at:source-dest-key-lcaf'" {
        description
            "When LCAF type type is Source/Dest.";
    }
    description
        "Source/Dest LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10#section-4.11";
    leaf source {
        type simple-address;
        description
            "Source address.";
    }
    leaf dest {
        type simple-address;
        description
```

```
        "Destination address.";
    }
}
container key-value-address {
    when "../address-type = 'lisp-at:key-value-address-lcaf'" {
        description
            "When LCAF type type is Key/Value Address.";
    }
    description
        "Key/Value Address LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10#section-4.11";
    leaf key {
        type simple-address;
        description
            "Address as Key.";
    }
    leaf value {
        type simple-address;
        description
            "Address as Value.";
    }
}
container service-path {
    when "../address-type = 'lisp-at:service-path-lcaf'" {
        description
            "When LCAF type service path identifier.";
    }
    description
        "Service Path LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ermagan-lisp-nsh-00";
    leaf service-path-id {
        type service-path-id-type;
        description
            "Service path identifier for the path for NSH header";
    }
    leaf service-index {
        type uint8;
        description
            "Service path index for NSH header";
    }
}
}
}
}
<CODE ENDS>
```

7.2. Data Model examples

This section presents some simple and illustrative examples on how to configure LISP.

7.2.1. LISP protocol instance

The following is an example configuration for a LISP protocol instance with the name "LISP1". There are also 2 VNIs configured.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <network-instances
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
    <network-instance>
      <name>VRF-BLUE</name>
      <vrf-root/>
      <enabled>true</enabled>
    </network-instance>
    <network-instance>
      <name>VRF-RED</name>
      <vrf-root/>
      <enabled>true</enabled>
    </network-instance>
  </network-instances>
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
          lisp:lisp
        </type>
        <name>LISP1</name>
        <lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
          <lisp-role>
            <lisp-role-type>etr</lisp-role-type>
          </lisp-role>
          <lisp-role>
            <lisp-role-type>itr</lisp-role-type>
          </lisp-role>
          <vpns>
            <vpn>
              <instance-id>1000</instance-id>
              <iid-name>VRF-BLUE</iid-name>
            </vpn>
            <vpn>
              <instance-id>2000</instance-id>
              <iid-name>VRF-RED</iid-name>
            </vpn>
          </vpns>
        </lisp>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>
```

7.2.2. LISP ITR

The following is an example configuration for ITR functionality under "LISP1". There are 2 Map-Resolvers configured.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
          lisp:lisp
        </type>
        <name>LISP1</name>
        <lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
          <lisp-role>
            <lisp-role-type>itr</lisp-role-type>
          </lisp-role>
          <itr xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp-itr">
            <map-resolvers>
              <map-resolver>2001:db8:203:0:113::1</map-resolver>
              <map-resolver>2001:db8:204:0:113::1</map-resolver>
            </map-resolvers>
          </itr>
        </lisp>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>
```

7.2.3. LISP ETR

The following is an example configuration for ETR functionality under "LISP1". There are 2 Map-Servers and 2 local EIDs configured.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <network-instances
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
    <network-instance>
      <name>VRF-BLUE</name>
      <vrf-root/>
      <enabled>true</enabled>
    </network-instance>
```

```
<network-instance>
  <name>VRF-RED</name>
  <vrf-root/>
  <enabled>true</enabled>
</network-instance>
</network-instances>
<routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
  <control-plane-protocols>
    <control-plane-protocol>
      <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
        lisp:lisp
      </type>
      <name>LISP1</name>
      <lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
        <lisp-role>
          <lisp-role-type>etr</lisp-role-type>
        </lisp-role>
        <lisp-router-id>
          <site-id>1</site-id>
        </lisp-router-id>
        <vpns>
          <vpn>
            <instance-id>1000</instance-id>
            <iid-name>VRF-BLUE</iid-name>
          </vpn>
          <vpn>
            <instance-id>2000</instance-id>
            <iid-name>VRF-RED</iid-name>
          </vpn>
        </vpns>
        <etr xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp-etr">
          <map-servers>
            <map-server>
              <ms-address>2001:db8:203:0:113::1</ms-address>
              <authentication-keys>
                <authentication-key>
                  <auth-key-id>key1</auth-key-id>
                  <auth-algorithm-id>
                    hmac-sha-256-128
                  </auth-algorithm-id>
                  <auth-key-value>*Kye^$$1#gb91U04zpa</auth-key-value>
                </authentication-key>
              </authentication-keys>
            </map-server>
            <map-server>
              <ms-address>2001:db8:204:0:113::1</ms-address>
              <authentication-keys>
                <authentication-key>
```



```
        <auth-key-id>key1</auth-key-id>
        <auth-algorithm-id>
            hmac-sha-256-128
        </auth-algorithm-id>
        <auth-key-value>*Kye^$$1#gb91U04zpa</auth-key-value>
    </authentication-key>
</authentication-keys>
</map-server>
</map-servers>
<local-eids>
    <vpn>
        <instance-id>1000</instance-id>
        <eids>
            <local-eid>
                <id>2001:db8:400:0:100::0</id>
                <eid-address>
                    <address-type xmlns:lisp-at=
                        "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
                        lisp-at:ipv6-prefix-afi
                    </address-type>
                    <ipv6-prefix>2001:db8:400:0:100::/80</ipv6-prefix>
                </eid-address>
            </local-eid>
        </eids>
    </vpn>
    <vpn>
        <instance-id>2000</instance-id>
        <eids>
            <local-eid>
                <id>2001:db8:800:0:200::0</id>
                <eid-address>
                    <address-type xmlns:lisp-at=
                        "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
                        lisp-at:ipv6-prefix-afi
                    </address-type>
                    <ipv6-prefix>2001:db8:800:0:200::/80</ipv6-prefix>
                </eid-address>
            </local-eid>
        </eids>
    </vpn>
</local-eids>
</etr>
</lisp>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</config>
```

7.2.4. LISP Map-Server

The following is an example configuration for Map-Server functionality under "LISP1". There are 2 mappings configured.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
          lisp:lisp
        </type>
        <name>LISP1</name>
        <lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
          <lisp-role>
            <lisp-role-type xmlns:lisp-ms=
              "urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver">
              lisp-ms:ms
            </lisp-role-type>
          </lisp-role>
          <map-server xmlns=
            "urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver">
            <sites>
              <site>
                <site-id>1</site-id>
                <authentication-keys>
                  <authentication-key>
                    <auth-key-id>key1</auth-key-id>
                    <auth-algorithm-id>
                      hmac-sha-256-128
                    </auth-algorithm-id>
                    <auth-key-value>*Kye^$$1#gb91U04zpa</auth-key-value>
                  </authentication-key>
                </authentication-keys>
              </site>
            </sites>
            <vpns>
              <vpn>
                <instance-id>1000</instance-id>
                <mappings>
                  <mapping>
                    <eid-id>1</eid-id>
                    <eid-address>
                      <address-type xmlns:lisp-at=
                        "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
                        lisp-at:ipv6-prefix-afi
```

```

        </address-type>
        <ipv6-prefix>2001:db8:400:0:100::/80</ipv6-prefix>
    </eid-address>
</mapping>
</mappings>
</vpn>
<vpn>
    <instance-id>2000</instance-id>
    <mappings>
        <mapping>
            <eid-id>1</eid-id>
            <eid-address>
                <address-type xmlns:lisp-at=
"urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
                    lisp-at:ipv6-prefix-afi
                </address-type>
                <ipv6-prefix>2001:db8:800:0:200::/80</ipv6-prefix>
            </eid-address>
        </mapping>
    </mappings>
</vpn>
</vpns>
</map-server>
</lisp>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</config>

```

8. Acknowledgments

The tree view and the YANG model shown in this document have been formatted with the 'pyang' tool.

9. IANA Considerations

The IANA is requested to assign a new namespace URI from the IETF XML registry.

This document registers the following namespace URIs in the IETF XML registry [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-lisp

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-itr

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-etr

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-mapresolver

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-address-types

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

This document registers the following YANG modules in the "YANG Module Names" registry [RFC6020]:

Name: ietf-lisp

Namespace: urn:ietf:params:xml:ns:yang:ietf-lisp

Prefix: lisp

Reference: RFC XXX

Name: ietf-lisp-itr

Namespace: urn:ietf:params:xml:ns:yang:ietf-lisp-itr

Prefix: lisp-itr

Reference: RFC XXX

Name: ietf-lisp-etr

Namespace: urn:ietf:params:xml:ns:yang:ietf-lisp-etr

Prefix: lisp-etr

Reference: RFC XXX

Name: ietf-lisp-mapserver

Namespace: urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver

Prefix: lisp-ms

Reference: RFC XXX

Name: ietf-lisp-mapresolver

Namespace: urn:ietf:params:xml:ns:yang:ietf-lisp-mapresolver

Prefix: lisp-mr

Reference: RFC XXX

Name: ietf-lisp-address-types

Namespace: urn:ietf:params:xml:ns:yang:ietf-lisp-address-types

Prefix: lisp-at

Reference: RFC XXX

10. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS

[RFC8446].

The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a pre-configured subset of all available NETCONF or RESTCONF protocol operations and content.

The security considerations of LISP control-plane [RFC6833] and LISP data-plane [RFC6830] as well as the LISP threat analysis [RFC7835] apply to this YANG model.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/  
lisp:lisp/
```

Access to the locator-sets node may modify which interfaces are used for data and/or control traffic as well as affect the load balancing of data-plane traffic. Access to the lisp-role node may prevent the device from performing its intended data-plane and/or control-plane operation. Access to the router-id node allows to modify the unique identifier of the device, which may result in disruption of its LISP control-plane operation. Access to the vpn node may allow to redirect data-plane traffic to erroneous local or remote network instances.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:map-server
```

Access to the sites node can prevent authorized devices from registering mappings in the Map-Server and/or allow unauthorized devices to do so. Access to the vpn node can result in corrupted mapping state that may propagate across the LISP network, potentially resulting in forwarding of data-plane traffic to arbitrary destinations and general disruption of the data-plane operation. Access to mapping-system-type and/or ddt-mapping-system nodes may prevent the device from connecting to the Mapping System infrastructure and consequentially to attract Map-Request messages.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:map-resolver
```

Access to mapping-system-type, ms-address and/or ddt-mapping-system nodes may prevent the device to connect to the Mapping System infrastructure and forward Map-Request messages.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp/lisp:itr
```

Access to the rloc-probing node can increase the control-plane overhead in the device or affect the capability of the device to detect failures on the underlay. Access to the itr-rlocs node may prevent the device from getting Map-Reply messages. Access to the map-resolvers node can prevent the device from sending its Map-Request messages to valid Map-Resolvers. Access to the proxy-etr nodes can affect the capability of the device to send data-plane traffic towards non-LISP destinations. Access to the map-cache node can result in forwarding of data-plane traffic to arbitrary destinations and general disruption of data-plane operation.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp/lisp:etr
```

Access to the map-servers node can prevent the device from registering its local mappings into the Mapping System. Access to the local-eids node can disrupt data-plane operation on the device and/or result in the device registering corrupted mappings into the Mapping System.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp
```

Access to the locator-sets node can expose the locators the device is using for its control and/or data operation. Access to the lisp-role node can disclose the LISP roles instantiated at the device which facilitates mounting attacks against the device. Access to the router-id node can expose the unique identifier of device which may allow a third party to track its control-plane operation and/or impersonate the device. Access to the vpn node can leak the local mapping between LISP Instance IDs and local network instances.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp/lisp:map-server
```


Access to the sites node can expose the credentials used to register mappings and allow unauthorized devices to do so. Access to the vpn node can expose the mappings currently registered in the device, which has privacy implications. Access to the mapping-system-type node may reveal the Mapping System in use which can be used to mount attacks against the device and/or the Mapping System. Access to the summary and counters nodes may expose operational statistics of the device.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:map-resolver
```

Access to the mapping-system-type node may reveal the Mapping System in use which can be used to mount attacks against the device and/or the Mapping System. Access to the ms-address and/or ddt-mapping-system nodes can leak the information about the Mapping System infrastructure used by the device, which can be used to block communication and/or mount attacks against it.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:itr
```

Access to the rloc-probing node can expose if and how the device is using control-plane signaling to probe underlay locators. Access to the itr-rlocs node may disclose the addresses the device is using to receive Map-Reply messages. Access to the map-resolvers node can expose the Map-Resolvers used by the device, which can be used to mount attacks against the device and/or the Mapping System. Access to the proxy-etrns node can disclose the PETRs used by the device, which can be used to mount attacks against the device and/or PETRs. Access to the map-cache node can expose the mappings currently cached in the device, which has privacy implications.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:etr
```

Access to the map-servers node can expose the credentials used by the device to register mappings into the Mapping System allowing an unauthorized device to impersonate and register mappings on behalf the authorized device. Access to the local-eids node can expose the local EIDs currently being served by the device, which has privacy implications.

11. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <<https://www.rfc-editor.org/info/rfc6832>>.
- [RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, DOI 10.17487/RFC6833, January 2013, <<https://www.rfc-editor.org/info/rfc6833>>.
- [RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, DOI 10.17487/RFC6836, January 2013, <<https://www.rfc-editor.org/info/rfc6836>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.

- [RFC7835] Saucez, D., Iannone, L., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Threat Analysis", RFC 7835, DOI 10.17487/RFC7835, April 2016, <<https://www.rfc-editor.org/info/rfc7835>>.
- [RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, DOI 10.17487/RFC8022, November 2016, <<https://www.rfc-editor.org/info/rfc8022>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.
- [RFC8111] Fuller, V., Lewis, D., Ermagan, V., Jain, A., and A. Smirnov, "Locator/ID Separation Protocol Delegated Database Tree (LISP-DDT)", RFC 8111, DOI 10.17487/RFC8111, May 2017, <<https://www.rfc-editor.org/info/rfc8111>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

[RFC8529] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Data Model for Network Instances", RFC 8529, DOI 10.17487/RFC8529, March 2019, <<https://www.rfc-editor.org/info/rfc8529>>.

Authors' Addresses

Vina Ermagan
Google
United States of America
Email: ermagan@gmail.com

Alberto Rodriguez-Natal
Cisco Systems
San Jose, CA
United States of America
Email: natal@cisco.com

Florin Coras
Cisco Systems
San Jose, CA
United States of America
Email: fcoras@cisco.com

Carl Moberg
Avassa
Email: calle@avassa.io

Reshad Rahman
Canada
Email: reshad@yahoo.com

Albert Cabellos-Aparicio
Technical University of Catalonia
Barcelona
Spain
Email: acabello@ac.upc.edu

Fabio Maino
Cisco Systems
San Jose, CA
United States of America

Email: fmaino@cisco.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: July 22, 2022

V. Moreno
Google LLC
D. Farinacci
lispers.net
A. Rodriguez-Natal
M. Portoles-Comeras
F. Maino
S. Hooda
Cisco Systems
January 18, 2022

Uberlay Interconnection of Multiple LISP overlays
draft-moreno-lisp-uberlay-05

Abstract

This document describes the use of the Locator/ID Separation Protocol (LISP) to interconnect multiple disparate and independent network overlays by using a transit overlay. The transit overlay is referred to as the "uberlay" and provides connectivity and control plane abstraction between different overlays. Each network overlay may use different control and data plane approaches and may be managed by a different organization. Structuring the network into multiple network overlays enables the interworking of different overlay approaches to data and control plane methods. The different network overlays are autonomous from a control and data plane perspective, this in turn enables failure survivability across overlay domains. This document specifies the mechanisms and procedures for the distribution of control plane information across overlay sites and in the uberlay as well as the lookup and forwarding procedures for unicast and multicast traffic within and across overlays. The specification also defines the procedures to support inter-overlay mobility of EIDs and their integration with the intra-overlay EID mobility procedures defined in draft-ietf-lisp-eid-mobility.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 22, 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Definition of Terms	3
3. Interconnecting multiple LISP site-overlays via the Uberlay .	4
3.1. Logical Topology Considerations	7
4. General Procedures	9
4.1. Control Plane Procedures	10
4.1.1. Split-horizon at the Border xTRs	11
4.1.2. Border-xTR Resiliency	12
4.2. Resolution and Forwarding Procedures	12
4.2.1. Multi-overlay requests at border xTR	13
4.3. Default EID registration and treatment	14
5. Multicast Specific Procedures	15
5.1. Inter-site-overlay Control Plane Procedures for Signal-free multicast	15
5.2. Border xTR Resolution and Forwarding procedures for Signal-free multicast	16
6. Inter site-overlay Mobility Procedures	16
7. Virtual Private Network (VPN) Considerations	18
8. IANA Considerations	18

9. Acknowledgements	18
10. References	18
10.1. Normative References	18
10.2. Informative References	19
Authors' Addresses	21

1. Introduction

The main motivation for this specification is to provide a methodology for the interconnection of LISP domains that may use disparate control and/or data plane approaches. For instance, one domain may use native LISP encapsulation for its data plane and a DDT based mapping system, while another domain may use VXLAN-GPE encapsulation and a mapping system based on [I-D.farinacci-lisp-decent]. Furthermore, one domain may use an IPv4 RLOC space and the other domain may use an IPv6 RLOC space and there may not be connectivity between the domains at the RLOC level. We propose a method to interconnect and enable interoperability between these disparate LISP overlay networks by connecting them to a common transit LISP overlay.

In order to provide interworking across implementations of overlays that may use different control and data plane approaches, a LISP network may be structured as a collection of site-overlays interconnected by a transit area. Each site-overlay is a fully functional overlay network and has its own set of Map Servers and Map Resolvers. Site-overlays share a border xTR with a transit area. Connectivity between site-overlays is provided via the transit area which we will refer to as "The Uberlay". This specification describes the Control Plane and Forwarding procedures for the implementation of an Uberlay connected multi-overlay LISP network. This approach to the structure of a LISP network may also enable regional failure survivability and fault isolation.

2. Definition of Terms

LISP related terms, notably Map-Request, Map-Reply, Ingress Tunnel Router (ITR), Egress Tunnel Router (ETR), Map-Server (MS) and Map-Resolver (MR) are defined in the LISP specification [RFC6830].

Terms defining interactions with the LISP Mapping System are defined in [RFC6833].

Terms related to the procedures for signal free multicast are defined in [RFC8378].

The following terms are here defined to facilitate the descriptions and discussions within this particular document.

Site-Overlay - Overlay network at a specific area or domain. This overlay network has a dedicated Mapping System.

Border-xTR - xTR that connects a site-overlay to one or more uberlays.

xTR - LISP Tunnel Router as defined in [RFC6833]. An xTR connects end-points to the site-overlay.

Local Mapping System - Mapping system of the site-overlay

Uberlay - Overlay network that interconnects different site-overlays to each other. The Uberlay has a dedicated Mapping System and creates an overlay amongst the border xTRs which connect different site-overlays.

Uberlay Mapping System - Autonomous mapping system dedicated to the uberlay.

Site-Overlay EIDs - Also referred to as local site-overlay EIDs, these are the EIDs that are connected to xTRs in a particular site-overlay and are registered in their own local site-overlay mapping system. These EIDs will also be registered in the uberlay but not in the remote site-overlay mapping systems.

Remote site-overlay EIDs - These are EIDs connected and registered in site-overlays other than the local site-overlay.

Local site-overlay EIDs - These are EIDs connected and registered in the local site-overlay.

3. Interconnecting multiple LISP site-overlays via the Uberlay

A LISP network can be structured as a collection of LISP site-overlays that are interconnected by one or more LISP Uberlays.

A LISP site-overlay is an overlay network that has its own set of xTRs, its own dedicated Mapping System and it may have a dedicated RLOC space, separate from that of other site-overlays or the uberlay. A LISP uberlay is also an overlay network with its own set of xTRs, its own dedicated Mapping System and it may have its own dedicated RLOC space. When the RLOC spaces are dedicated, RLOC routes in the local underlay do not leak across to the underlay of other site-overlays.

A site-overlay will have xTRs and Border xTRs. The xTRs provide connectivity to the local site-overlay EIDs, which are the EIDs that are locally connected to the overlay-site. The Border xTRs are Re-

encapsulating Tunnel Routers (RTRs) that connect the site-overlays to the LISP Uberlay in the transit network. xTRs participate in one site-overlay and one site-overlay only. Border xTRs participate in the mapping system of the site-overlay it resides in and the mapping system of the uberlay it connects the site-overlay to. Border xTRs may be shared by more than one site-overlay.

The different site-overlays can be interconnected by an uberlay. The uberlay consists of a dedicated Mapping System and the set of Border xTRs that connect the participating site-overlays to the Uberlay and the Uberlay Mapping System.

Each site-overlay will have its own set of Map Servers and Map Resolvers (MS/MRs) which operate as an autonomous Mapping System. The Uberlay Mapping System is also autonomous and includes all necessary Map Servers and Map Resolvers. Any of the Mapping Systems, in site-overlays or in the Uberlay, follow the control plane specification in [RFC6833] and may be structured as a Distributed Delegation Tree (DDT) per [RFC8111] for the purposes of horizontal scaling or other optimizations within each Mapping System.

The MS/MRs can be co-located with the border-xTRs of the site-overlay. When a Border xTR services more than one site-overlay, and the MS/MRs are instantiated on the Border xTR, logical instances of MS/MRs must be dedicated to each site-overlay.

This specification defines the interaction between the Mapping Systems of the site-overlays and the uberlay to deliver a multi-overlay hierarchical network. The forwarding procedures relevant to the border xTRs are also specified. Figure 1 illustrates the multi-overlay network.

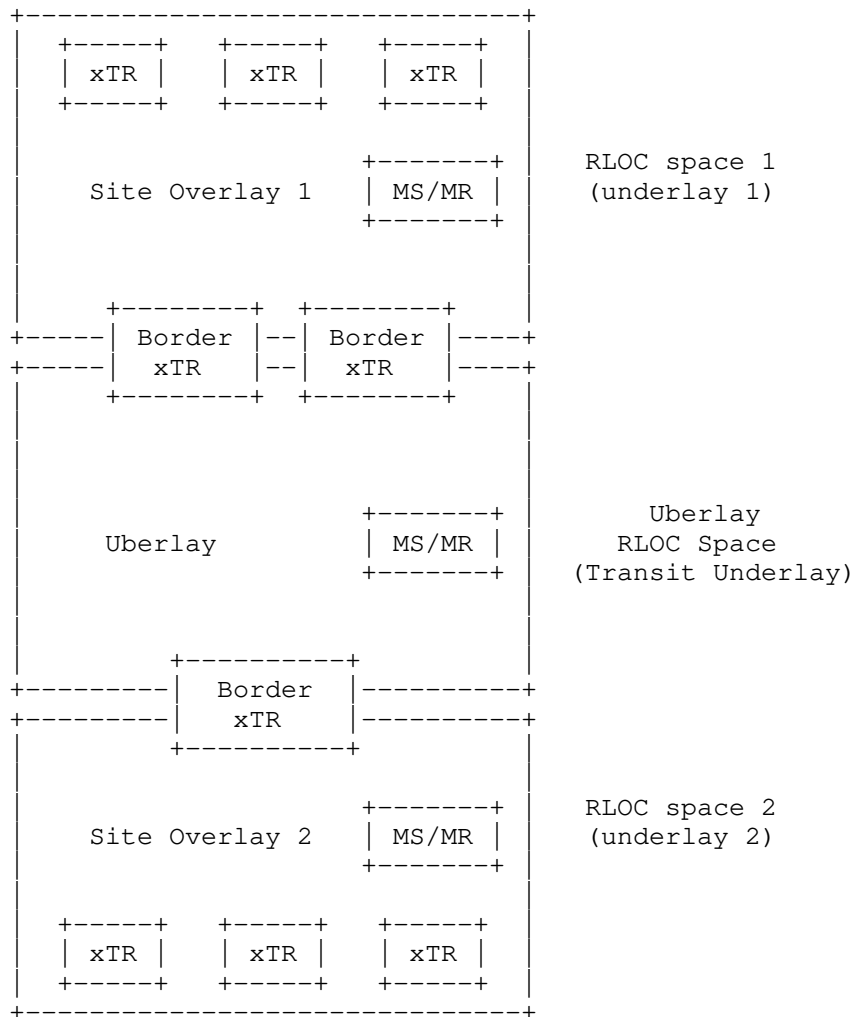


Figure 1. Site-overlays connected via Uberlay

Structuring the LISP network as multiple site-overlays interconnected by an uberlay delivers the following benefits:

- o Enable the interworking of diverse site-overlay implementations in which different mapping systems and encapsulations may be used.

- o Enhanced resiliency through regional failure survivability. Failures in one site-overlay or failures in a portion of the underlay should not affect other site-overlays.
- o Reduce the state of the site-overlay control plane. The site-overlay control plane will only maintain state for EIDs that are connected to xTRs within the site-overlay. These EIDs are referred to as local site-overlay EIDs in this specification. Remote site-overlay EIDs will not be explicitly registered within the site-overlay.
- o Separate the RLOC space of the different site-overlays as well as the uberlay RLOC space. Each site-overlay will only need reachability to its own RLOCs, making the RLOCs private to the site-overlay. Similarly, the uberlay RLOC space does not require knowledge of site-overlay specific RLOCs. This simplifies the underlay routing protocol structure and reduces the state that must be handled and maintained by the underlay routing protocols.
- o Reduced latency for local site-overlay EID registrations may be achieved when xTRs and Map Servers are topologically close. Topological proximity is expected when the RLOC spaces for the different overlays are kept separate.
- o Reduced latency for local site-overlay EID lookups may be achieved when xTRs, Map Resolvers and Map Servers are topologically close. Topological proximity is expected when the RLOC spaces for the different overlays are kept separate.
- o Creates a multicast replication hierarchy where the Border RTRs serve as the points of multicast replication for multicast traffic that spans multiple site-overlays.
- o Creates a distributed structure of RTRs that can be leveraged for the deployment of NAT traversal in the RLOC space.
- o

3.1. Logical Topology Considerations

xTRs as defined in RFC6833bis connect a network to the LISP overlay and register the EID prefixes from the connected network to the LISP mapping system. Border xTRs, as defined in this document, will connect site-overlays to the Uberlay and register the EID prefixes that originate in a site-overlay in the Mapping System of the Uberlay. Conversely, a border xTR may register EID prefixes present in the Uberlay Mapping System into the Mapping System of a particular site-overlay. Furthermore, border xTRs may connect Uberlays to each

other and register the EID prefixes from one Uberlay into the other. There is no provision for the detection of registration loops when concatenating site-overlays and Uberlays, thus any interconnection of overlay domains (site-overlays or Uberlays) must be done in a loop free topology.

A loop free topology is hereby defined for reference. This is a general concept and is not encoded into any of the protocol messages in LISP. A loop free topology limits the peerings between Uberlays and/or overlays to a strict hierarchy. At the top of the hierarchy is a single central Uberlay or Core Uberlay. The loop free topology is defined by two simple rules: Uberlays must only connect to Uberlays in the next consecutive level of hierarchy (no level skipping) and uberlays within the same level of hierarchy must not connect to each other. The loop-free topology hierarchy is illustrated in Figure 2.

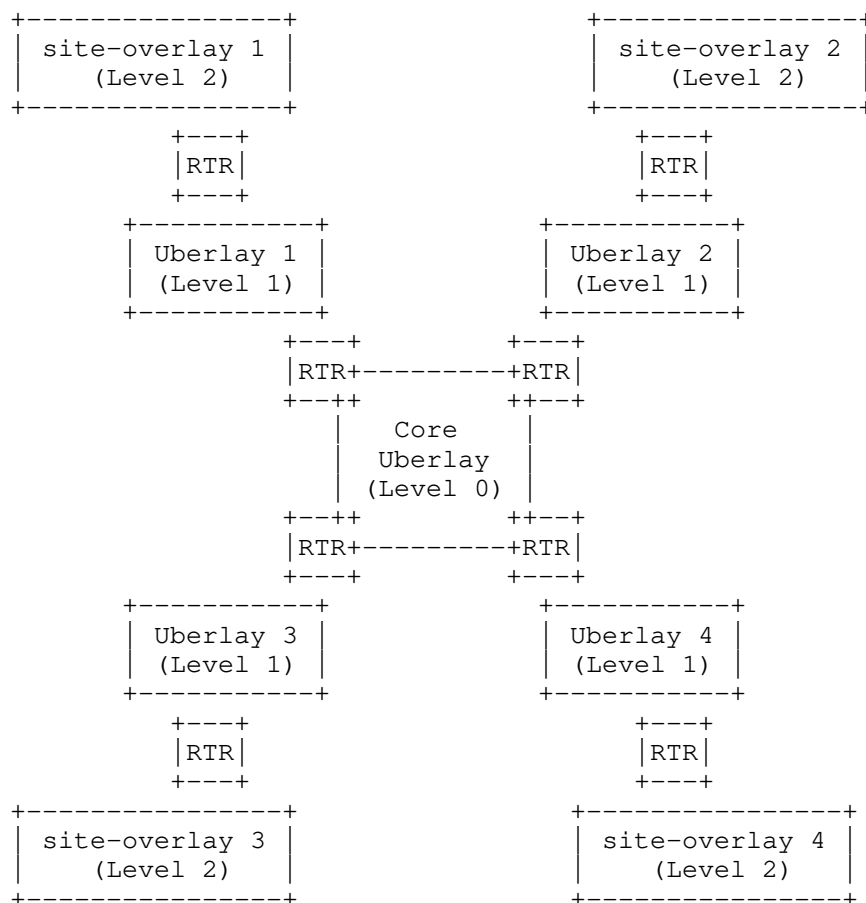


Figure 2. Loop-free topology hierarchy

4. General Procedures

A site-overlay maintains state only for its local site-overlay EIDs and RLOCs. Tunnels never cross site-overlay or uberlay boundaries. Remote site-overlay EIDs are reachable at the source site-overlay via a default mapping which will steer all traffic destined to remote site-overlay EIDs to the border xTRs where it can be handed off to the uberlay. Traffic will be decapsulated at the border xTRs and a lookup in the uberlay mapping system will determine the site-overlay to which traffic is to be re-encapsulated. The uberlay maintains state for the EIDs of all interconnected site-overlays and will steer traffic from the source site-overlay to the destination site-overlay by encapsulating the traffic from the source site-overlay border xTR

to the destination site-overlay border xTR. At the border xTR of the destination site-overlay, traffic will be de-capsulated, a lookup in the local destination site-overlay Mapping System will take place and traffic will be re-encapsulated to the xTR that connects to the destination EID. Thus, forwarding is achieved by concatenating overlays and doing Re-encapsulation at the border xTRs to forward the traffic from the Ingress site-overlay to the Egress site-overlay via the Uberlay.

Traffic for non-LISP sites, or for EIDs not registered in any site-overlay, will also be forwarded to the border xTR where it will be forwarded or dropped as appropriate.

4.1. Control Plane Procedures

Local EIDs must be registered by the xTRs into the local Mapping System of the site-overlay. Intra-site communication follows the standard procedures of registration, resolution, caching and encapsulation defined in [I-D.ietf-lisp-rfc6830bis] and [I-D.ietf-lisp-rfc6833bis] amongst the xTRs within the local site-overlay.

The border xTRs at a site-overlay should have a local site-overlay RLOC-set and will also have an uberlay RLOC-set. The local site-overlay RLOC-set is in the private site-overlay RLOC space and is used by the border xTRs as the RLOC set for any mappings it may register with the site-overlay Mapping System. The uberlay RLOC-set for the border-xTRs of a particular site-overlay are the RLOCs to reach the site-overlay in the uberlay RLOC space. The border xTR will use the uberlay RLOC-set in any mappings it may register with the uberlay Mapping System. It is possible for a deployment to connect the RLOC spaces of the site-overlays and the uberlay, it is also possible in the scenario of a common RLOC space for the uberlay and local site-overlay RLOC sets to be one and the same. Any implementation of this specification should support disjoint RLOC spaces or joint RLOC spaces.

The border xTRs must register a default EID-prefix as specified in Section 4.3 with the local site-overlay Mapping System. Remote EIDs will be generally reachable by xTRs in a site-overlay using the default EID mapping registered by the border xTRs. This is expected to be the mapping used for most communications to remote site-overlay EIDs. Remote site-overlay EIDs may be registered with the local site-overlay Mapping System for the purposes of supporting inter-overlay EID mobility as specified in Section 6, these mappings will be preferred over the default EID mapping whenever present.

Local EIDs registered with the site-overlay mapping system must also be registered with the Uberlay Mapping System. The registration of the local site-overlay EIDs with the uberlay Mapping System is originated by the Border xTRs. The local site-overlay EIDs SHOULD be aggregated into the shortest covering prefix possible before being registered with the uberlay Mapping System. How this aggregation is achieved is implementation specific.

In order to be able to register the local site-overlay EIDs with the uberlay Mapping System, the border xTRs must subscribe to all EIDs registered in their local site-overlay Mapping System. This is a subscription to 0.0.0.0/0 (or 0::/0) with the N-bit set as specified in [I-D.ietf-lisp-pubsub]. The subscription populates all local site-overlay EID mappings in the map-cache of the border xTRs.

Once received through the subscription, the local site-overlay EIDs in the map-cache at the border xTRs must be registered by the border xTRs with the uberlay Mapping System. The local site-overlay EIDs will be registered using the 'uberlay' RLOC-set for the registering border xTR.

Following [I-D.ietf-lisp-eid-mobility], the border xTRs will also subscribe to any EID prefixes it registers with the uberlay Mapping System. This allows the border xTRs to get Map Notify messages from the uberlay Mapping System for EID prefixes that may move from their local site-overlay to a remote site-overlay.

4.1.1. Split-horizon at the Border xTRs

Remote site-overlay EIDs may be learnt at a border xTR due to resolution of a remote destination EID or due to a mobility event as specified in Section 6. Remote site-overlay EIDs learnt from the uberlay will be installed in the map-cache of the border xTR with the corresponding remote uberlay RLOC-set for the remote border xTR. When these remote site-overlay EIDs are learnt as a consequence of the map-notify messages defined in the Inter-overlay mobility procedures in Section 6, the EIDs will also be registered with the local site-overlay mapping system using the local site-overlay RLOC-set for the border-xTR. The remote site-overlay EIDs registered with the local site-overlay mapping system will be learnt back at the border xTR because of the border xTR's subscription to all local site-overlay EIDs. This can cause the mapping for the remote EID that is installed in the border xTR map-cache to flip flop between the uberlay RLOC-set and the local site-overlay RLOC-set.

In order to avoid this flip flopping a split horizon procedure must be implemented. When a mapping received at the border xTR (as part of its subscription to all local site-overlay EID prefixes) has the

local site-overlay RLOC-set for the border xTR, the mapping received in the subscription corresponds to a remote site-overlay EID and should be ignored by the border xTR. The mapping should not be installed in the map-cache of the border xTR and the EIDs in the mapping should not be advertised to the uberlay. More robust split horizon mechanisms can be proposed in future revisions of this specification.

4.1.2. Border-xTR Resiliency

Redundancy at the border xTRs requires that border xTRs be logically grouped so that the redundant array doesn't create a registration loop. As border xTRs interconnect overlay domains, the border xTRs will register the EID prefixes from one domain into the neighboring domain. From the perspective of the border xTR, the EID prefixes to be registered in one domain are learnt from a neighbor domain which we will refer to as the "site-of-origin". The site-of-origin may be an overlay-site, an Uberlay or an IP network.

Border xTRs should be logically grouped in Border Sets. A border set is a group of border xTRs that register EID prefixes from the same site-of-origin. Members of a border set will register the EIDs from a particular site-of-origin into the neighboring overlay (site-overlay or uberlay) using a common site-id. The use of the site-ID namespace is locally significant to each overlay domain (site-overlay or Uberlay) and does not require cross-domain synchronization or dispersion. A border-xTR may be a member of multiple border sets to allow different site-of-origin domains to be serviced by the border-xTR. Note that not all site-of-origin domains will connect to the same combination of border-xTRs.

EID Mappings will be tagged with a site-ID according to their site-of-origin when they are registered by the border-xTR. The site-ID must be maintained in the Mapping System as part of the registration record. EID Mappings published and received at the border xTR must include the site-ID for the EID Mapping. If the border-xTR receives a mapping for an EID with a site-ID that matches the site-ID for one of its border sets (site-of-origin), the Border xTR will not register that information to the site-of-origin associated with that site-ID and thus prevent any registration loops from occurring.

4.2. Resolution and Forwarding Procedures

Intra-site communication follows the standard procedures of registration, resolution, caching and encapsulation defined in [I-D.ietf-lisp-rfc6830bis] and [I-D.ietf-lisp-rfc6833bis] amongst the xTRs within the local site-overlay.

Inter-site communication is achieved by encapsulating traffic destined to remote site-overlay EIDs from the xTRs to the border xTRs. Traffic will be decapsulated at the border xTRs and a lookup in the uberlay mapping system will determine the site-overlay to which traffic is to be re-encapsulated. The lookup should return the uberlay RLOCs for the border xTRs of the site-overlay where the destination EID is located. At the border xTR of the destination overlay-site, traffic will be de-capsulated, and re-encapsulated to the destination xTR, just like an RTR does. The border xTR already has the destination EID in its cache per its subscription to all local site-overlay EIDs.

When receiving encapsulated traffic, a border xTR will de-capsulate the traffic and will do a lookup for the destination EID in its map cache. If the destination EID is present in the map cache, the traffic is forwarded and no lookup takes place. If the destination EID is not present in the cache, the destination EID is not in any local site-overlay connected to the border xTR, in which case the border xTR will issue a map-request to all Uberlay Mapping Systems it is connected to. The criteria to determine which Mapping Systems are Uberlay Mapping Systems is simply to select those Mapping Systems with which the border xTR doesn't hold a subscription to 0.0.0.0/0 (or 0::/0).

4.2.1. Multi-overlay requests at border xTR

A Border xTR may query all Mapping Systems in all uberlays it participates in. The border xTR will then chose based on longest prefix match the more specific EID mapping provided by any of the Mapping Systems. This procedure could also include site-overlay Mapping Systems, however those are not expected to be queried as the border xTR subscribes to all EIDs in the site-overlays and the presence of the mappings in the cache will prevent any lookups. The processing of Map Requests following the multi-domain request logic works as follows:

1. The Border xTR sends a map request for the prefix that it intends to resolve to each of the uberlay Mapping Systems it participates in.
2. The Border xTR receives Map Replies from each of the different uberlay Mapping Systems it sent requests to. The Border xTR will treat the replies differently depending on their contents:
 - * Negative Map Replies (NMR) are ignored and discarded unless all Map Replies received are Negative, then the border xTR follows the procedures specified in [RFC6833] for Negative Map Replies.

- * Map Replies with RLOCs that belong to the requesting border xTR are ignored.
- * Map Replies with EID prefixes that are not already in the map cache of the border xTR are accepted and cached.
- * If the EID prefix received in the Map-Reply already exists in the cache/routing table, but the Map-Reply contains a different RLOC-set than the one cached, the mappings are merged so that the RLOCs received in the Map-Reply are added to the RLOC-set previously cached for the EID prefix.
- * If the EID prefix received in the Map-Reply is more specific or less specific than an EID prefix already cached, the mapping received MUST be cached.

It is expected that a deployment of the uberlay would include the dynamic registration of default EIDs. It is also recommended that an implementation adopts mechanisms for the dynamic resolution of default EIDs. In an environment leveraging the dynamic registration and resolution of default EIDs, the border xTR should not receive Negative Map-Replies, but all replies (including those in response to requests for destinations that are external to the EID space) will be Map-replies with a non-zero locator count. Nevertheless, an implementation could opt to not use dynamic default-EID handling. In these cases, the border xTR will receive NMRs. The implementation of the Border xTR should defer the decision on caching an NMR until all relevant Map-replies are received. To this effect, the implementation should implement mechanisms to ensure that sufficient replies are received before programming the map-cache. The mechanisms by which this is achieved are an implementation specific matter and therefore not specified in this document.

When following these rules to process multi-domain requests, the Border xTR guarantees proper discovery and use of destination prefixes that will be associated with their corresponding overlay-site. By ignoring the negative replies the procedure works regardless of whether the Mapping Systems of multiple uberlays have consistent configurations or operate individually without being aware of the whole addressing space in the overlay fabric.

4.3. Default EID registration and treatment

Border xTRs will register a mapping to be used as a default mapping to handle the forwarding of traffic destined to any EIDs that are not explicitly registered. These mappings will be registered in the local site-overlay Mapping System of each site-overlay. The RLOCs for the mappings will be the site-overlay RLOCs of the border xTR.

This registration is intended to instruct the Mapping System to follow the procedures in [RFC6833] for Negative Map Replies and calculate the broadest non-registered EID prefix that includes the requested destination EID and issue a map-reply with the calculated EID and the RLOCs registered by the border xTRs. The map-reply for this default mapping will have a shorter TTL to accommodate any changes in the registrations.

The instruction to the Mapping System can be encoded as the registration of an agreed upon distinguished name such as "Default". The registration will contain the RLOC set desired for the default handling.

5. Multicast Specific Procedures

This specification will focus on the procedures necessary to extend signal-free multicast [RFC8378] across multiple site-overlays interconnected with an uberlay. The specification will focus on the extensions of the Sender and Receiver site procedures

5.1. Inter-site-overlay Control Plane Procedures for Signal-free multicast

1. At the listener sites, xTRs with multicast listeners will follow the receiver site procedures described in [RFC8378]. A replication list will be built and registered on the site-overlay Mapping System for the multicast channel being joined by the listeners.
2. The Mapping System for the listener site-overlay will send Map-Notify messages towards the multicast source or RP per [RFC8378]. The multicast source or RP is reachable via the border-xTRs of the listener site-overlay via the default EID mapping registered in the listener site-overlay.
3. Upon reception of the Map-Notify in the previous step, the listener site-overlay border-xTR will register the multicast EID with the uberlay Mapping System using the uberlay RLOCs for its site-overlay as the RLOC set for the mapping being registered. Only one of the RLOCs in the set should be active in the registration per the procedures in [RFC8378]. A replication tree is built in the uberlay as specified in [RFC8378].
4. After the listener site-overlay border-xTR registers the multicast EID with the uberlay Mapping system, the uberlay MS will send a Map-Notify toward the multicast source per [RFC8378]

5. Upon reception of the Map-Notify in the previous step, the border xTR at the source site-overlay registers its interest in the multicast EID with the source site-overlay Mapping System following the procedures described in [RFC8378].

5.2. Border xTR Resolution and Forwarding procedures for Signal-free multicast

The mapping resolution procedures for multicast EIDs at border xTRs fall within the scope of the mechanisms specified in Section 4. The Map-replies obtained from the lookup will follow the behavior specified in [RFC8378] for signal-free multicast.

Forwarding will also follow the General Procedures specified in Section 4 without alteration. It is worth noting that the concatenation of overlays between listener sites, uberlay and sender site-overlays creates a convenient replication structure where the border xTRs act as the replication points to form an optimized end-to-end multi-level replication tree.

6. Inter site-overlay Mobility Procedures

The receiver and sender site procedures defined in [I-D.ietf-lisp-eid-mobility] apply without change to each site-overlay and to the uberlay. Border xTRs are connected to two or more overlay networks which are following the mobility procedures. An away table is defined at the border xTR for each overlay network it participates in. In order to illustrate the procedures required, this specification describes a scenario where a border xTR has one local site-overlay away table and one uberlay facing away table. The procedures for mobility described in this section are extensible to border xTRs participating in more than two overlays.

When a map notify for an EID is received at an xTR, an away entry is created on the receiving side table. Any away entries for the specific EID in other tables on the same LISP node (xTR or RTR) must be removed. This general rule addresses convergence necessary for a first move as well as any subsequent moves (moves that take place after the away tables are already populated with entries for the moving EID due to previous moves).

The following set of procedures highlights any additions to the mobility procedures defined in [I-D.ietf-lisp-eid-mobility]:

1. Detect the roaming EID per the mechanisms described in [I-D.ietf-lisp-eid-mobility] and register the EID with the site-overlay Mapping System at the landing site-overlay

2. The site-overlay Mapping System at the landing site-overlay must send a Map-Notify to the last registrant xTR (if it is local to the site-overlay) and to the border xTR as the border xTR subscribes to all EIDs in the site-overlay.
3. The border xTR will install an entry for the moved host in the local away table of the border xTR.
4. The border xTR from the landing site-overlay will register the roaming EID with the uberlay Mapping System using the uberlay RLOC-set for the landing site-overlay
5. The Uberlay Map Server will send Map-Notify messages to the border xTRs at the departure site-overlay as specified in [I-D.ietf-lisp-eid-mobility] (border xTR with the previously registered RLOCs).
6. Upon reception of the Map-Notify, the border xTR must check if the Map-Notify is for an EID-prefix that is covered by a broader or equal EID-prefix that is locally registered. Local registration is determined by the presence of the broader or equal EID prefix in the map-cache of the border xTR.
7. If the roaming EID-prefix received in the Map-Notify is not covered under a previously registered EID-prefix in the local site-overlay, the EID-prefix is a newly registered prefix and no further action is required.
8. If the roaming EID-prefix received in the Map-Notify is covered under a registered EID-prefix, the Map-Notify is due to a move event. In this case, the site-overlay border xTR must register the roaming EID prefix in the site-overlay mapping system using the site-overlay facing RLOC-set of the border-xTRs. The roaming EID-prefix must also be installed in the uberlay facing away table of the border xTR at the departure site-overlay.
9. The departure site-overlay Map-Server will send Map-Notify messages to the xTRs at the departure site-overlay as specified in [I-D.ietf-lisp-eid-mobility] (edge xTRs with the previously registered RLOCs).
10. When the site-overlay xTR at the departure site-overlay receives the Map-Notify from the border xTR, it will include the EID prefix received in the Map-Notify in its away table per the procedures described in [I-D.ietf-lisp-eid-mobility].
11. Data triggered Solicit Map Requests (SMRs) will be initiated in the different site-overlays and the uberlay as traffic matches

the different away tables. As specified in [I-D.ietf-lisp-eid-mobility], these SMRs notify the different ITRs involved in communications with the roaming EID that they must issue a new Map-Request to the mapping system to renew their mappings for the roaming EID.

7. Virtual Private Network (VPN) Considerations

When supporting multiple Instance IDs as specified in [I-D.ietf-lisp-vpn] the Instance IDs range is divided in two sets. A reuse-set that can be used in each site-overlay and a global-set used across site-overlays and the uberlay.

Instance-IDs that are local to a site-overlay should only provide intra-overlay connectivity and are in the site-overlay mapping system only for VPN use for the xTRs in the site-overlay. When the VPN reaches across site-overlays, then the global-set instance-IDs are in the uberlay mapping system as well as each site-overlay mapping system where the VPN members exist.

8. IANA Considerations

This document has no IANA implications

9. Acknowledgements

The authors want to thank Kedar Karamarkar, Prakash Jain and Vina Ermagan for their insightful contribution to shaping the ideas in this document. We would also like to acknowledge the valuable input from the workgroup chairs Joel Halpern and Luigi Iannone in refining the objectives of the document.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3618] Fenner, B., Ed. and D. Meyer, Ed., "Multicast Source Discovery Protocol (MSDP)", RFC 3618, DOI 10.17487/RFC3618, October 2003, <<https://www.rfc-editor.org/info/rfc3618>>.

- [RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, DOI 10.17487/RFC4601, August 2006, <<https://www.rfc-editor.org/info/rfc4601>>.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <<https://www.rfc-editor.org/info/rfc4607>>.

10.2. Informative References

- [I-D.farinacci-lisp-decent]
Farinacci, D. and C. Cantrell, "A Decent LISP Mapping System (LISP-Decent)", draft-farinacci-lisp-decent-08 (work in progress), August 2021.
- [I-D.ietf-lisp-eid-mobility]
Comeras, M. P., Ashtaputre, V., Moreno, V., Maino, F., and D. Farinacci, "LISP L2/L3 EID Mobility Using a Unified Control Plane", draft-ietf-lisp-eid-mobility-08 (work in progress), July 2021.
- [I-D.ietf-lisp-pubsub]
Rodriguez-Natal, A., Ermagan, V., Cabellos, A., Barkai, S., and M. Boucadair, "Publish/Subscribe Functionality for LISP", draft-ietf-lisp-pubsub-09 (work in progress), June 2021.
- [I-D.ietf-lisp-rfc6830bis]
Farinacci, D., Fuller, V., Meyer, D., Lewis, D., and A. Cabellos, "The Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-rfc6830bis-36 (work in progress), November 2020.
- [I-D.ietf-lisp-rfc6833bis]
Farinacci, D., Maino, F., Fuller, V., and A. Cabellos, "Locator/ID Separation Protocol (LISP) Control-Plane", draft-ietf-lisp-rfc6833bis-30 (work in progress), November 2020.
- [I-D.ietf-lisp-vpn]
Moreno, V. and D. Farinacci, "LISP Virtual Private Networks (VPNs)", draft-ietf-lisp-vpn-08 (work in progress), January 2022.

- [RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, DOI 10.17487/RFC6407, October 2011, <<https://www.rfc-editor.org/info/rfc6407>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC6831] Farinacci, D., Meyer, D., Zwiebel, J., and S. Venaas, "The Locator/ID Separation Protocol (LISP) for Multicast Environments", RFC 6831, DOI 10.17487/RFC6831, January 2013, <<https://www.rfc-editor.org/info/rfc6831>>.
- [RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, DOI 10.17487/RFC6833, January 2013, <<https://www.rfc-editor.org/info/rfc6833>>.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.
- [RFC8061] Farinacci, D. and B. Weis, "Locator/ID Separation Protocol (LISP) Data-Plane Confidentiality", RFC 8061, DOI 10.17487/RFC8061, February 2017, <<https://www.rfc-editor.org/info/rfc8061>>.
- [RFC8111] Fuller, V., Lewis, D., Ermagan, V., Jain, A., and A. Smirnov, "Locator/ID Separation Protocol Delegated Database Tree (LISP-DDT)", RFC 8111, DOI 10.17487/RFC8111, May 2017, <<https://www.rfc-editor.org/info/rfc8111>>.
- [RFC8378] Moreno, V. and D. Farinacci, "Signal-Free Locator/ID Separation Protocol (LISP) Multicast", RFC 8378, DOI 10.17487/RFC8378, May 2018, <<https://www.rfc-editor.org/info/rfc8378>>.

Authors' Addresses

Victor Moreno
Google LLC
1600 Amphitheater Parkway
Mountain View, CA 94043
USA

Email: vimoreno@google.com

Dino Farinacci
lispers.net
San Jose, CA 95120
USA

Email: farinacci@gmail.com

Alberto Rodriguez-Natal
Cisco Systems
170 Tasman Drive
San Jose, California 95134
USA

Email: natal@cisco.com

Marc Portoles-Comeras
Cisco Systems
170 Tasman Drive
San Jose, California 95134
USA

Email: mportole@cisco.com

Fabio Maino
Cisco Systems
170 Tasman Drive
San Jose, California 95134
USA

Email: fmaino@cisco.com

Sanjay Hooda
Cisco Systems
170 Tasman Drive
San Jose, California 95134
USA

Email: shooda@cisco.com