

LISP Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 17, 2013

L. Cheng  
M. Sun  
ZTE Corporation  
July 16, 2012

draft-cheng-lisp-shdht-01  
LISP Single-Hop DHT Mapping Overlay

Abstract

This draft specifies the LISP Single-Hop Distributed Hash Table Mapping Overlay (LISP-SHDHT), a distributed mapping database which embodies SHDHT Nodes to maintain (Key, value) pairs for LISP (Locator/ID Separation Protocol)-like architecture, wherein every (key value) pair is according to an EID(Endpoint ID)-to-RLLOC(Routing Locator) mapping information entry. According to this strategy, EID is hashed to be a unique Resource ID which is used for locating destiny DHT Node who maintains mapping entry for the particular EID. Furthermore, adaptive hash space partition method is adopted to solve the load balance problem on SHDHT Nodes which is common on traditional DHT planes.

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 <http://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 January 17, 2013.

Copyright Notice

Copyright (c) 2012 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 (<http://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 . . . . .	4
3. SHDHT Overview . . . . .	4
3.1. Node ID and Partition ID . . . . .	4
3.2. Data Storage and Hash Assignment . . . . .	6
3.3. Node Routing Table . . . . .	6
4. LISP SHDHT . . . . .	7
4.1. SHDHT Node Operation . . . . .	7
4.2. ITR Operation . . . . .	8
4.3. ETR Operation . . . . .	8
4.4. Encapsulated Message Format . . . . .	9
4.4.1. Encapsulated Map Request . . . . .	9
5. Mobility Considerations . . . . .	10
6. Security Considerations . . . . .	10
7. IANA Considerations . . . . .	10
8. References . . . . .	10
8.1. Normative References . . . . .	10
8.2. Informational References . . . . .	11
Authors' Addresses . . . . .	11

## 1. Introduction

Locator/ID Separation Protocol (LISP) [I-D.ietf-lisp] specifies an architecture and mechanism for replacing the address currently used by IP with two separate name spaces: Endpoint IDs (EIDs), used within LISP sites, and Routing Locators (RLOCs), used on transit networks that make up the Internet infrastructure. To achieve this separation, LISP defines protocol mechanisms for mapping from EIDs to RLOCs. As a result, an efficient database is needed to store and propagate those mappings globally. Several such mapping databases have been proposed, among them: LISP-NERD [I-D.lear-lisp-nerd], LISP-ALT [I-D.ietf-lisp-alt], LISP-DDT [I-D.fuller-lisp-ddt], and LISP-DHT [I-D.fuller-lisp-ddt].

According to hybrid model databases such like LISP-ALT [I-D.ietf-lisp-alt] and LISP-DDT [I-D.fuller-lisp-ddt], architectures of these mapping databases are based on announcement/delegation of hierarchically-delegated segments of EID namespace (i.e., prefixes). Therefore, based on these architectures, when a roaming event occurs and a LISP site or a LISP MN receives new RLOCs, the site or MN has to anchor pre-configured map-server to register its new mapping information no matter where the site or MN currently locates, just in order to protect EID prefixes announced aggregately in the database [I-D.meyer-lisp-mn].

As a DHT strategy based mapping database, LISP-DHT [I-D.mathy-lisp-dht] exhibits several interesting properties, such as self-configuration, self-maintenance, scalability and robustness that are clearly desirable for a EID-to-RLOC resolution service. However, this database is based on multi-hop Chord DHT. On one hand, inquiries of mapping information in this case need to pass through iterative multi-hop lookup steps which will cause relatively large delay time. On the other hand, load balance between Chord nodes is another essential problem need to be solved.

This draft specifies a Single-Hop Distributed Hash Table Mapping Overlay (LISP-SHDHT) which provides mapping information lookup service for sites running LISP. Main characters of this strategy is that,

1. Each SHDHT Node maintains routing information for all other SHDHT Nodes. Thus, messages interaction between SHDHT Nodes in the same SHDHT overlay just need one or two hops.
2. Traditionally, Node IDs are used to identify DHT nodes and represent hash space arrangement on DHT nodes. In SHDHT strategy, the two roles are separated. Partition IDs are adopted for hash space arrangement and a build-in load balancing solution

is designed.

This draft specifies the outline of SHDHT and the basic application of LISP SHDHT. In actual deployment of LISP SHDHT, mapping database could be maintained by operators and could be deployed as collaborative combination of multiple domain LISP SHDHTs. Deployment of collaborative domain LISP SHDHTs is for future study, as well as the security strategies on/between domain LISP SHDHTs.

## 2. Definition of Terms

This draft uses terms defined in [I-D.ietf-lisp]. This section defines some new terms used in this document.

SHDHT: Single-Hop Distributed Hash Table Mapping Overlay.

SHDHT Node: Physical nodes which compose SHDHT overlay's topology.

Node ID: Node identifier, which is used for maintenance. Each SHDHT Node has a unique Node ID. The ring containing Node IDs indicates overlay's topology.

Partition ID: Partition identifier, which is used for hash space assignment. Partition IDs and Resource IDs share the same hash space. All Partition IDs in overlay are unique. Each SHDHT Node could have multiple Partition IDs. The ring containing Partition IDs determines how the hash space is partitioned into segments and how these segments are assigned to nodes.

Resource ID: Each data object stored in DHT overlay could be hashed to be a unique Resource ID. In LISP-SHDHT strategy, data objects are according to the EIDs. Resource IDs share the same hash space with Partition IDs. As a result, SHDHT Nodes perform data objects put/get/remove operations based on these IDs.

Node Routing Table: Routing table on SHDHT Nodes.

## 3. SHDHT Overview

### 3.1. Node ID and Partition ID

Most of existing DHTs use node IDs for both maintenance and hash space arrangement. For example, in LISP-DHT[I-D.mathy-lisp-dht], each chord node of the DHT ring has a unique k-bits identifier (ChordID). Nodes perform operations such like put/get/remove based on ChordIDs. Furthermore, ChordIDs are also used to associate nodes

with hash space segments that the nodes responsible for.

In SHDHT, two roles of maintenance and hash space arrangement are separated and a new kind identifier called Partition ID is adopted. Each SHDHT node has a unique Node ID which identifies the physical node and multiple Partition IDs which represent hash space segments. All Partition IDs in the overlay are also unique. Node IDs and Partition IDs are mapped into two ring-shaped spaces respectively. The ring containing Node IDs indicates the overlay's topology. The ring containing Partition IDs determines how the hash space is partitioned into segments and how these segments are assigned to nodes. It is noteworthy that SHDHT Nodes could determine number of Partition IDs on them separately and could generate Partition IDs randomly just need to make sure that the generated Partition IDs will not conflict with existing Partition IDs on the SHDHT plane.

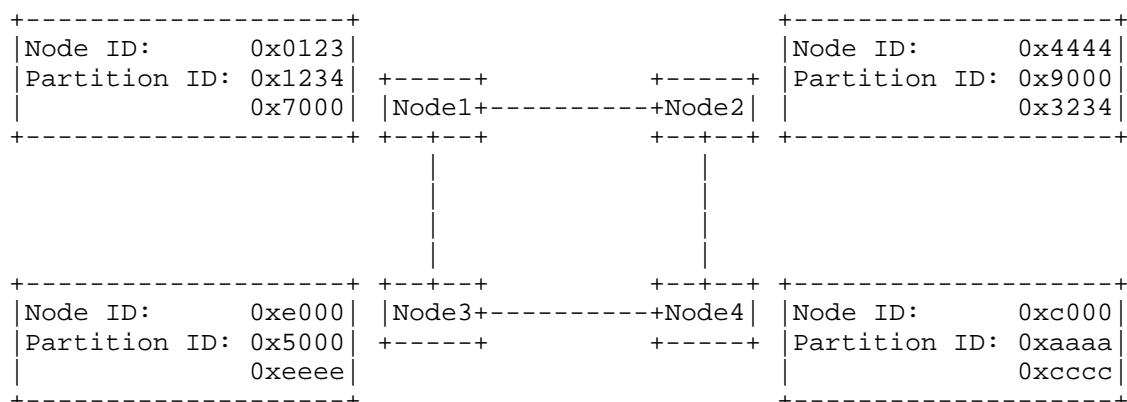


Fig.1 SHDHT Deployment Example

As shown in Fig.1 is an example of SHDHT. This SHDHT overlay is consist of four SHDHT NODEs each has a unique Node ID and maintains two Partition IDs. According to this deployment, hash space is partitioned to be eight segments each is indexed by a Partition ID. From Fig. I, it could be observed that hash space segments are not required to be partitioned equally. As SHDHT Nodes could general Partition IDs separately, when a SHDHT Node gets all hash segments assignment information for other SHDHT Nodes, it will be able to implement the load balance of SHDHT overlay by general proper Partition IDs.

In SHDHT, each SHDHT Node stores and maintains data objects. Data objects are indexed by Resource IDs which share the same hash space with Partition IDs and will locate in the hash space segments whose Partition IDs are closest to their Resource IDs.

For example, for a data object whose Resource ID is 0x8213, the Resource ID locates between Partition ID 0x7000 and Partition ID 0x9000. As Partition ID 0x9000 is closer to Resource ID 0x8213, the data object will be stored and maintained on Node2 who is assigned with the hash space segment indexed by Partition ID 0x9000.

### 3.2. Data Storage and Hash Assignment

In traditional DHTs, hash space is partitioned into segments based on node IDs. As a result, data objects are always stored in their root nodes, whose node IDs are "closest" to data objects' Resource IDs. Replications of data objects in a particular node are always stored in the preceding node or successor node of the root node. The backup preceding node or successor node will automatically become the new closest node if the root node leaves the overlay.

In SHDHT, the whole hash space is partitioned into segments based on partition IDs. The root node of a data object is the node, which has the closest partition ID to the data object's Resource ID. In SHDHT, each node can maintain multiple hash space segments with respective Partition IDs. As the preceding Partition ID or successor Partition ID may be owned by the same root node. Replication of data objects could still be stored in preceding node or successor node of root node.

### 3.3. Node Routing Table

In SHDHT, each node maintains a Node Routing Table containing routing information for all other SHDHT Nodes locate in the same SHDHT overlay. Table I shows the Node Routing Table on SHDHT Nodes of Fig.1. A Node Routing Table contains all Partition IDs and their associated Node IDs and node addresses. For simplification, Node IDs and Partition IDs shown in the draft are only 16-bit numbers.

When SHDHT Node receives a message points to a particular Resource ID, it could look up Node Routing Table and find out the Partition ID which is closest to the Resource ID. Furthermore, message could be transferred to the corresponding SHDHT Node.

Partition ID	Node ID	Address
0x1234	0x0123	10.0.0.2:2000
0x3234	0x4444	10.0.0.3:2000
0x5000	0xe000	10.0.0.4:2000
0x7000	0x0123	10.0.0.2:2000
0x9000	0x4444	10.0.0.3:2000
0xaaaa	0xc000	10.0.0.5:2000
0xcccc	0xc000	10.0.0.5:2000
0xeeee	0xe000	10.0.0.4:2000

TABLE II SHDHT Node Routing Table

For example, if Node 1 (ID: 0x1234) in Fig.1 needs to implement put/get/remove operations for a data object with Resource ID 0x8213. Node 1 first lookups its Node Routing Table and finds out that the closest Partition ID to this Resource ID is 0x9000. Then Node 1 will send put/get/remove request to the node owns the Partiton ID, in Fig.1 is Node2, whose Node ID is 0x4444 and address is 10.0.0.3:2000.

#### 4. LISP SHDHT

LISP SHDHT is proposed to provide "EID-to-RLOC(s)" mapping information lookup service for sites running the Locator/ID Separation Protocol (LISP).

In SHDHT, all EID-to-RLOC mapping entries are stored in SHDHT Nodes as data objects. Each SHDHT Node have a RLOC address. EIDs in mapping entries can be hashed as Resource IDs of data objects. All SHDHT Nodes in the same SHDHT overly perform hash operation based on the same hash algorithm.

Data objects stored in LISP SHDHT Nodes may be in the following format:

```
Object (lisp) = [EID prefix, (RLOC1, priority, weight),
...,RLOCn, priority, weight), TTL ]
```

##### 4.1. SHDHT Node Operation

In SHDHT, each SHDHT Node plays the roles of mapping server and forwarding router. When a SHDHT Node receives a control message from the LISP data plane, it will perform the following operations,

1. SHDHT Node extracts destination EID address from the control message.
2. SHDHT Node map the EID address to be Resource ID based on the shared hash algorithm.
3. SHDHT Node look up the Node Routing Table and find out the Partition ID which is closest to the Resource ID.
4. If the Resource ID locates in hash space segments the SHDHT Node responsible for, SHDHT Node decapsulates the control message and performs proper operations according to message type:
  - \* If the message is a Map-Request, SHDHT Node first checks if there's data objection, i.e. corresponding mapping entry, matching the Resource ID. If there is no match, the SHDHT Node returns a encapsulated negative Map-Reply. If there is matched mapping entry, the SHDHT Node returns the encapsulated Map-Reply. Furthermore, if the queried EID is covered by an EID prefix mapping entry, SHDHT Node could choose to return EID prefix mapping as response.
  - \* If the message is a Map-Register, SHDHT Node extracts and stores mapping information in the message.
5. Otherwise, SHDHT Node re-encapsulates the control message and sends it to the corresponding node based on routing information in Node Routing Table.

#### 4.2. ITR Operation

According to LISP-MS [I-D.ietf-lisp-ms], LISP ITRs use Map Resolvers as proxy to send control messages, such like encapsulated Map-Requests and Map-Replies.

In scenario of LISP SHDHT, SHDHT Nodes could also be configured as a Map Resolvers for the ITRs. An ITR could send encapsulated Map-Requests to a SHDHT Node directly. If the selected SHDHT Node is the node who maintains the matching information, it will respond Map-Replies directly. Otherwise, it will forward Map-Requests into the DHT overlay, and forward Map-Replies to ITRs when it receives responds from other SHDHT Nodes.

#### 4.3. ETR Operation

According to LISP-MS [I-D.ietf-lisp-ms], LISP ETRs register mapping information onto the Map Server by sending Map-Register messages.



In scenario of LISP SHDHT, SHDHT Nodes play the role of Map Server. Thus, encapsulated Map-Register messages should be sent to corresponding SHDHT Nodes. As a result, ETRs could send Map-Register messages to a nearest SHDHT Node who will forward messages to the corresponding SHDHT Nodes who should take on the responsibility.

#### 4.4. Encapsulated Message Format

An Encapsulated Control Message (ECM) defined in [I-D.ietf-lisp] is used to encapsulate control packets sent between xTRs and mapping database system. At this time, only Map-Request messages are allowed to be encapsulated in ECM format. When ITRs choose a SHDHT Node as proxy to send control messages, they could use encapsulated message format defined in , as shown in Fig.2.

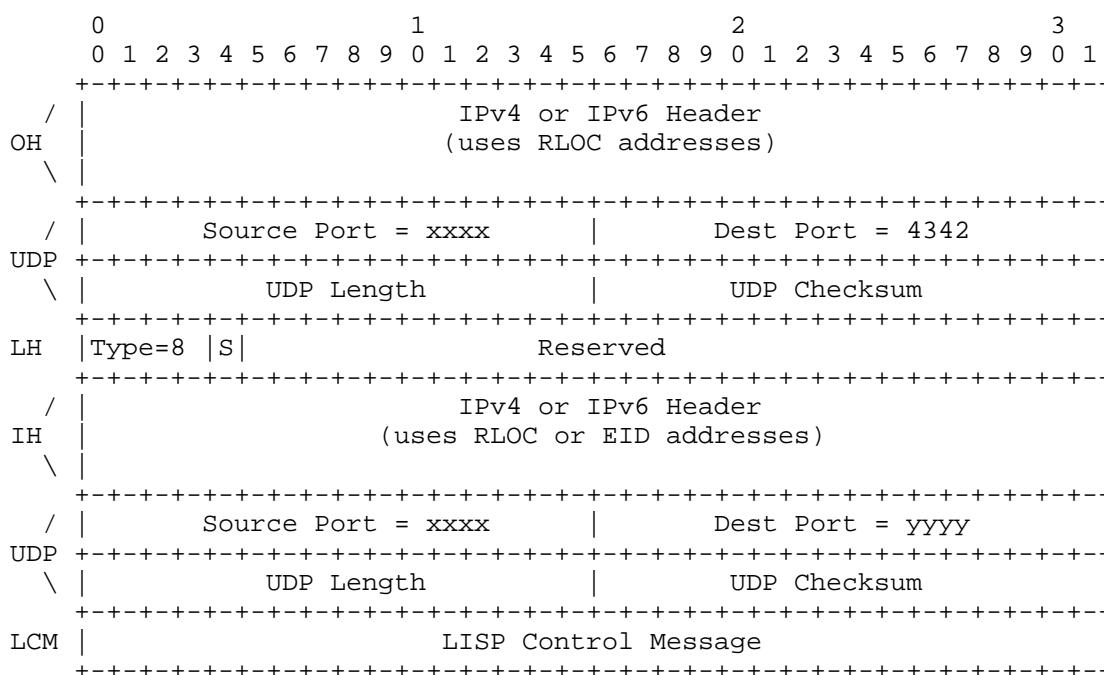


Fig.2 Encapsulated Control Message Format

##### 4.4.1. Encapsulated Map Request

Suppose that the selected SHDHT Node of an ITR is Node1.

When the ITR sends Encapsulated Map-Requests to Node1, source address and destination address in message OH (Outside Header) should be RLOC addresses of ITR and Node1 respectively.

In the IH (Inside Header), source address is still RLOC address of Nodel, while destination address is the inquired EID address.

Consider Nodel is a configured Map Resolver, and then the configuration of Encapsulated Map-Request message has not been changed.

## 5. Mobility Considerations

As specified in section 4.2 and 4.3, ITR/ETR could choose a nearest LISP SHDHT Node as proxy to send control messages.

Based on LISP SHDHT, when roaming events occurs, the roamed LISP sites or LISP MNs are no longer need to anchor pre-configured map-servers.

## 6. Security Considerations

TBD

## 7. IANA Considerations

This document makes no requests to IANA.

## 8. References

### 8.1. Normative References

[I-D.fuller-lisp-ddt]

Fuller, V., Lewis, D., and V. Ermagan, "LISP Delegated Database Tree", March 2012.

[I-D.ietf-lisp]

Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol (LISP)", May 2012.

[I-D.ietf-lisp-alt]

Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "LISP Alternative Topology (LISP+ALT)", December 2011.

[I-D.mathy-lisp-dht]

Mathy, L., Iannone, L., and O. Bonaventure, "LISP-DHT: Towards a DHT to map identifiers onto locators  
<http://biblio.info.ucl.ac.be/2008/456949.txt>",

February 2008.

[I-D.meyer-lisp-mn]

Farinacci, D., Lewis, D., Meyer, D., and C. White, "LISP Mobile Node", April 2012.

## 8.2. Informational References

[I-D.ietf-lisp-ms]

Fuller, V. and D. Farinacci, "LISP Map Server Interface", March 2012.

[I-D.lear-lisp-nerd]

Lear, E., "NERD: A Not-so-novel EID to RLOC Database", April 2012.

## Authors' Addresses

Li Cheng  
ZTE Corporation  
R&D Building 1, Zijinghua Road No.68  
Nanjing, Yuhuatai District 210012  
P.R.China

Email: cheng.li2@zte.com.cn

Mo Sun  
ZTE Corporation  
R&D Building 1, Zijinghua Road No.68  
Nanjing, Yuhuatai District 210012  
P.R.China

Email: sun.mo@zte.com.cn



LISP Working Group  
Internet-Draft  
Intended status: Informational  
Expires: January 17, 2013

J. N. Chiappa  
Yorktown Museum of Asian Art  
July 16, 2012

An Architectural Perspective on the LISP  
Location-Identity Separation System  
draft-chiappa-lisp-architecture-01

Abstract

LISP upgrades the architecture of the IPvN internetworking system by separating location and identity, current intermingled in IPvN addresses. This is a change which has been identified by the IRTF as a critically necessary evolutionary architectural step for the Internet. In LISP, nodes have both a 'locator' (a name which says where in the network's connectivity structure the node is) and an 'identifier' (a name which serves only to provide a persistent handle for the node). A node may have more than one locator, or its locator may change over time (e.g. if the node is mobile), but it keeps the same identifier.

This document gives additional architectural insight into LISP, and considers a number of aspects of LISP from a high-level standpoint.

[NOTE: This is still a somewhat rough draft version; a few sections at the end are just rough frameworks, but almost all the key sections, and all the front part of the document, are here, and in something like reasonably complete form.]

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to format it for publication as an RFC or to translate it into languages other than English.

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 <http://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 January 17, 2013.

Copyright Notice

Copyright (c) 2012 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 (<http://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.	Goals of LISP
2.1.	Reduce DFZ Routing Table Size
2.2.	Deployment of New Namespaces
2.3.	Future Development of LISP
3.	Architectual Perspectives
3.1.	Another Packet-Switching Layer
3.2.	'Double-Ended' Approach
4.	Architectual Aspects
4.1.	Critical State
4.2.	Need for a Mapping System
4.3.	Piggybacking of Control on User Data
5.	Namespaces
5.1.	LISP EIDs
5.1.1.	Residual Location Functionality in EIDs
5.2.	RLOCs
5.3.	Overlapping Uses of Existing Namespaces
5.4.	LCAFs
6.	Scalability
6.1.	Demand Loading of Mappings
6.2.	Caching of Mappings
6.3.	Amount of State
6.4.	Scalability of The Indexing Subsystem
7.	Security
7.1.	Basic Philosophy
7.2.	Design Guidance
7.2.1.	Security Mechanism Complexity
7.3.	Security Overview
7.3.1.	Securing Lookups
7.3.2.	Securing The Indexing Subsystem
7.3.3.	Securing Mappings
7.4.	Securing the xTRs
8.	Robustness
9.	Fault Discovery/Handling
10.	Optimization
11.	Open Issues
11.1.	Local Open Issues
11.1.1.	Missing Mapping Packet Queueing
11.1.2.	Mapping Cache Management Algorithm
11.2.	Systemic Open Issues
11.2.1.	Mapping Database Provider Lock-in
11.2.2.	Automated ETR Synchronization
11.2.3.	EID Reachability
11.2.4.	Detect and Avoid Broken ETRs
12.	Acknowledgments
13.	IANA Considerations
14.	Security Considerations
15.	References
15.1.	Normative References
15.2.	Informative References
	Appendix A. Glossary/Definition of Terms
	Appendix B. Other Appendices

## 1. Introduction

This document begins by introducing some high-level architectural perspectives which have proven useful for thinking about the LISP location-identity separation system. It then discusses some architectural aspects of LISP (e.g. its namespaces). The balance (and bulk) of the document contains architectural analysis of the LISP system; that is, it reviews from a high-level standpoint various aspects of that system; e.g. its scalability, security, robustness, etc.

NOTE: This document assumes a fair degree of familiarity with LISP; in particular, the reader should have a good 'high-level' understanding of the overall LISP system architecture, such as is provided by [Introduction], "An Introduction to the LISP System".

By "system architecture" above, the restricted meaning used there is: 'How the system is broken up into subsystems, and how those subsystems interact; when does information flows from one to another, and what that information is.' There is obviously somewhat more to architecture (e.g. the namespaces of a system, in particular their syntax and semantics), and that remaining architectural content is covered here.

## 2. Goals of LISP

As previously stated in the abstract, broadly, the goal of LISP is to be a practically deployable architectural upgrade to IPvN which performs separation of location and identity. But what is the value of that? What will it allow us to do?

The answer to that obviously starts with the things mentioned in the "Initial Applications" section of [Introduction], but there are other, longer-range (and broader) goals as well.

### 2.1. Reduce DFZ Routing Table Size

One of the main design drivers for LISP, as well as other location-identity separation proposals, is to decrease the overhead of running global routing system. In fact, it was this aspect that led the IRTF Routing RG to conclude that separation of location and identity was a key architectural underpinning needed to control the growth of the global routing system. [RFC6115]

As noted in [Introduction], many of the practical needs of Internet users are today met with techniques that increase the load on the global routing system (Provider Independent addresses for the provision of provider independence, multihoming, etc; more-specific routes for TE; etc.) Provision of these capabilities by a mechanism which does not involve extra load on the global routing system is therefore very desirable.

A number of factors, including the use of these techniques, has led to a great increase in the fragmentation of the address space, at least in terms of routing table entries. In particular, the growth in demand for multi-homing has been forseen as driving a large increase in the size of the global routing tables.

In addition, as the IPv4 address space becomes fuller and fuller, there will be an inevitable tendency to find use in smaller and smaller 'chunks' of that space. [RFC6127] This too would tend to increase the size of the global routing table.

LISP, if successful and widely deployed, offers an opportunity to use separation of location and identity to control the growth of the size of the global routing table. (A full examination of this topic is beyond the scope of this document - see {{find reference}}.)

### 2.2. Deployment of New Namespaces

Once the mapping system is widely deployed and available, it should make deployment of new namespaces (in the sense of new syntax, if not new semantics) easier. E.g. if someone wishes in the future to devise a system which uses native MPLS [RFC3031] for a data carriage system joining together a large number of xTRs, it would easy enough to arrange to have the mappings for destinations attached to those xTRs abe some sort of MPLS-specific name.

More broadly, the existence of a binding layer, with support for multiple namespace built into the interface on both sides (see Section 5) is a tremendously powerful evolutionary tool; one can introduce a new namespace (on one side) more easily, if it is mapped to something which is already deployed (on the other). Then, having taken that step, one can invert the process, and deploy yet another new namespace, but this time on the other.

### 2.3. Future Development of LISP

Speculation about long-term future developments which are enabled by the deployment of LISP is not really proper for this document. However, interested readers may wish to consult [Future] for one person's thoughts on this topic.

### 3. Architectural Perspectives

This section contains some high-level architectural perspectives which have proven useful in a number of ways for thinking about LISP. For one, when trying to think of LISP as a complete system, they provide a conceptual structure which can aid analysis of LISP. For another, they can allow the application of past analysis of, and experience with, similar designs.

#### 3.1. Another Packet-Switching Layer

When considering the overall structure of the LISP system at a high level, it has proven most useful to think of it as another packet-switching layer, run on top of the original internet layer - much as the Internet first ran on top of the ARPANET.

All the functions that a normal packet switch has to undertake - such as ensuring that it can reach its neighbours, and they they are still up - the devices that make up the LISP overlay also have to do, along the 'tunnels' which connect them to other LISP devices.

There is, however, one big difference: the fanout of a typical LISP ITR will be much larger than most classic physical packet switches. (ITRs only need to be considered, as the LISP tunnels are all effectively unidirectional, from ITR to ETR - an ETR needs to keep no per-tunnel state, etc.)

LISP is, fundamentally, a 'tunnel' based system. Tunnel system designs do have their issues (e.g. the high inter-'switch' fan-out), but it's important to realize that they also can have advantages, some of which are listed below.

#### 3.2. 'Double-Ended' Approach

LISP may be thought of as a 'double-ended' approach to enhancing the architecture, in that it uses pairs of devices, one at each end of a communication stream. In particular, to interact with the population of 'legacy' hosts (which will be, inevitably, the vast majority, in the early stages of deployment) it requires a LISP device at both ends of the 'tunnel'.

This is in distinction to, say, NAT systems ([RFC1631]), which only need a device deployed at one end: the host at the other end doesn't need a matching device at its end to massage the packets, but can simply consume them on its own, as any packets it receives are fully normal packets. This allows any site which deploys such a 'single-ended' device to get the full benefit, whilst acting entirely on its own. [Wasserman]

The issue is not that LISP uses tunnels. Designs like HIP ([RFC4423]) and ILNP ([ILNP]), which do not involve tunnels, inhabit a similar space to tunnel-based designs like LISP, in that unless



both ends are upgraded - or there is a proxy at the un-upgraded end - one doesn't get any benefits. So it's really not the tunnel which is the key aspect, it's the 'all at one end' part which is key. Whether the system is tunnel, versus non-tunnel, is not that important.

However, the double-ended approach of LISP does have advantages, as well as costs. To put it simply, the 'feature' of the alternative approach, that there's only a box at one end, has a 'bug': there's only a box at one end. There are things which such a design cannot accomplish, because of that.

To put it another way, does the fact that the packet thus necessarily has only a single 'name' in it for the entities at each end (i.e. the IPvN source and destination addresses), because it is a 'normal' packet, present a limitation? Put that way, it would seem natural that it should cause certain limits.

To compile a complete list of the things that can be done, when two separate 'names' are in the packet, is beyond the scope of this document. However, one example of the kind of thing that can be done is mobility with open connections, without needing to 'triangle route' the packets through some sort of 'base station' at the original location. Another is that it is possible to automatically tunnel IPv6 traffic over IPv4 infrastructure, or vice versa, invisibly to the hosts on both ends.

In the longer term, having having tunnel boxes will allow (and is allowing) us to explore other kinds of wrappings. For example, we can transport 'raw' local-network packets (such as Ethernet MAC frames) across an IPvN infrastructure.

One could also wrap packets in non-IPvN formats: perhaps to take direct advantage of the capabilities of underlying switching fabrics (e.g. MPLS [RFC3031]); perhaps to deploy new carriage protocols, etc, where non-standard packet formats will allow extended semantics.

#### 4. Architectural Aspects

LISP does take some novel architectural approaches in a number of ways: e.g. its use of a separate mapping system, etc, etc. This section contains some commentary on some of the high-level architectural aspects of LISP.

##### 4.1. Critical State

LISP does have 'critical state' in the network (i.e. state which, if lost, causes the communication to fail). However, because LISP is designed as an overall system, 'designing it in' allows for a 'systems' approach to its state issues. In LISP, this state has been designed to be maintained in an 'architected' way, so it does not produce systemic brittleness in the way that the state in NATs does.

For instance, throughout the system, provisions have been made to have redundant copies of state, in multiple devices, so that the loss of any one device does not necessarily cause a failure of an ongoing connection.

##### 4.2. Need for a Mapping System

LISP does need to have a mapping system, which brings design, implementation, configuration and operational costs. Surely all these costs are a bad thing? However, having a mapping system have advantages, especially when there is a mapping layer which has global visibility (i.e. other entities know that it is there, and have an interface designed to be able to interact with it). This is unlike, say, the mappings in NAT, which are 'invisible' to the rest of the network.

In fact, one could argue that the mapping layer is LISP's greatest strength. Wheeler's Axiom\* ('Any problem in computer science can be solved with another level of indirection') indicates that the binding layer available with the LISP mapping system will be of great value. Again, it is not the job of this document to list them all - and in any event, there is no way to foresee them all.

The author of this document has often opined that the hallmark of great architecture is not how well it does the things it was designed to do, but how well it does things it was never expected to have to handle. Providing such a powerful and generic binding layer is one sure way to achieve the sort of lasting flexibility and power that leads to that outcome.

[Footnote \*: This Axiom is often mis-attributed to Butler Lampson, but Lampson himself indicated that it came from David Wheeler.]

#### 4.3. Piggybacking of Control on User Data

LISP piggybacks control transactions on top of user data packets. This is a technique that has a long history in data networking, going back to the early ARPANET. [McQuillan] It is now apparently regarded as a somewhat dubious technique, the feeling seemingly being that control and user data should be strictly segregated.

It should be noted that none of the piggybacking of control functionality in LISP is architecturally fundamental to LISP. All of the functions in LISP which are performed with piggybacking could be performed almost equally well with separate control packets.

The "almost" is solely because it would cause more overhead (i.e. control packets); neither the response time, robustness, etc would necessarily be affected - although for some functions, to match the response time observed using piggybacking on user data would need as much control traffic as user data traffic.

This technique is particularly important, however, because of the issue identified at the start of this section - the very large fanout of the typical LISP switch. Unlike a typical router, which will have control interactions with only a few neighbours, a LISP switch could eventually have control interactions with hundreds, or perhaps even thousands (for a large site) of neighbours.

Explicit control traffic, especially if good response times are desired, could amount to a very great deal of overhead in such a case.

#### 5. Namespaces

One of the key elements in any architecture, or architectural analysis, are the namespaces involved: what are their semantics and syntax, what are the kinds of things they name, etc.

LISP has two key namespace, EIDs and RLOCs, but it must be emphasized that on an architectural level, neither the syntax, or, to a lesser degree, the semantics, of either are absolutely fixed. There are certain core semantics which are generally unchanging (such as the notion that EIDs provide only identity, whereas RLOCs provide location), but as we will see, there is a certain amount of flexibility available for the long-term.

In particular, all of LISP's key interfaces always include an Address Family Identifier (AFI) [AFI] for all names, so that new forms can be introduced at any time the need is felt. Of course, in practise such an introduction would not be a trivial exercise - but neither is it impossibly painful, as is the case with IPv4's 32-bit addresses,

which are effectively impossible to upgrade.

#### 5.1. LISP EIDs

A 'classic' EID is defined as a subset of the possible namespaces for endpoints. [Chiappa] Like most 'proper' endpoint names, as proposed there, they contain contain no information about the location of the endpoint. EIDs are the subset of possible endpoint names which are: fixed length, 'reasonably' short', binary (i.e. not intended for direct human use), globally unique (in theory), and allocated in a top-down fashion (to achieve the former).

LISP EIDs are, in line with the general LISP deployment philosophy, a reuse of something already existing - i.e. IPvN addresses. For those used as in LISP as EIDs, LISP removes much (or, in some cases, all) of the location-naming function of IPvN addresses.

In addition, the goal is to have EIDs name hosts (or, more properly, their end-end communication stacks), whereas the other LISP namespace group (RLOCs) names interfaces. The idea is not just to have two namespaces (with different semantics), but also to use them to name different classes of things - classes which currently do not have clearly differentiated names. This should produce even more functionality.

##### 5.1.1. Residual Location Functionality in EIDs

LISP retains, especially in the early stages of the deployment, in many cases some residual location-naming functionality in EIDs. This is to allow the packet to be correctly routed/forwarded to the destination node, once it has been unwrapped by the ETR - and this is a direct result of LISP's deployment philosophy (see [Introduction], Section "Deployment").

Clearly, if there are one or more unmodified routers between the ETR and the destination node, those routers will have to perform a routing step on the packet, for which it will need some information as to the location of the destination.

One can thus view such LISP EIDs, which retain 'stub' location information, as 'addresses' (in the definition of the generic sense of this term, as used here), but with the location information restricted to a limited, local scope.

This retention of some location functionality in LISP EIDs, in some cases, has led some people to argue that use of the name 'EID' is improper. In response, it was suggested that LISP use the term 'LEID', to distinguish LISP's 'bastardized' EIDs from 'true' EIDs, but this usage has never caught on.

It has also been suggested that one usage mode for LISP EIDs, in existing software loads, is to assign them as the address on an internal virtual interface; all the real interfaces would have RLOCs only. [Templin] This would make such LISP EIDs functionally equivalent to 'real' EIDs - they are names which are purely identity, have no location information of any kind in them, and cannot be used to make any routing decisions anywhere outside the host.

It is true that even in such cases, the EID is still not a 'pure' EID, as it names an interface, not the end-end stack directly. However, to do a perfect job here (or on separation of location and identity) is impossible without modifying existing hosts (which are, inevitably, almost always one end of an end-end communication) - and that has been ruled out, for reasons of viable deployment.

The need for interoperation with existing unmodified hosts limits the semantic changes one can impose, much as one might like to provide a

cleaner separation. (Future evolution can bring us toward that state, however: see [Future].)

## 5.2. RLOCs

RLOCs are basically pure 'locators' [RFC1992], although their syntax and semantics is restricted at the moment, because in practise the only forms of RLOCs supported are IPv4 and IPv6.

## 5.3. Overlapping Uses of Existing Namespaces

It is in theory possible to have a block of IPvN namespace used as both EIDs and RLOCs. In other words, EIDs from that block might map to some other RLOCs, and that block might also appear in the DFZ as the locators of some other ETRs.

This is obviously potentially confusing - when a 'bare' IPvN address from one of these blocks, is it the RLOC, or the EID? Sometimes it is obvious from the context, but in general one could not simply have a (hypothetical) table which assigns all of the address space to either 'EID' or 'RLOC'.

In addition, such usage will not allow interoperation of the sites named by those EIDs with legacy sites, using the PITR mechanism ([Introduction], Section "Proxy Devices"), since that mechanism depends on advertizing the EIDs into the DFZ, although the LISP-NAT mechanism should still work ([Introduction], Section "LISP-NAT").

Nevertheless, as the IPv4 namespace becomes increasingly used up, this may be an increasingly attractive way of getting the 'absolute last drop' out of that space.

## 5.4. LCAFs

{{To be written.}}

--- Key-ID  
--- Instance-IDs

## 6. Scalability

As with robustness, any global communication system must be scalable, and scalable up to almost any size. As previously mentioned (xref target="Perspectives-Packet"/), the large fanouts to be seen with LISP, due to its 'overlay' nature, present a special challenge.

One likely saving grace is that as the Internet grows, most sites will likely only interact with a limited subset of the Internet; if nothing else, the separation of the world into language blocks means that content in, say, Chinese, will not be of interest to most of the rest of the world. This tendency will help with a lot of things which could be problematic if constant, full,  $N^2$  connectivity were likely on all nodes; for example the caching of mappings.

### 6.1. Demand Loading of Mappings

One question that many will have about LISP's design is 'why demand-load mappings - why not just load them all'? It is certainly true that with the growth of memory sizes, the size of the complete database is such that one could reasonably propose keeping the entire thing in each LISP device. (In fact, one proposed mapping system for LISP, named NERD, did just that. [NERD])

A 'pull'-based system was chosen over 'push' for several reasons; the main one being that the issue is not just the pure `_size_` of the mapping database, but its `_dynamicity_`. Depending on how often mappings change, the update rate of a complete database could be

relatively large.

It is especially important to realize that, depending on what (probably unforeseeable) uses eventually evolve for the identity->location mapping capability LISP provides, the update rate could be very high indeed. E.g. if LISP is used for mobility, that will greatly increase the update rate. Such a powerful and flexible tool is likely to be used in unforeseen ways (Section 4.2), so it's unwise to make a choice that would preclude any which raise the update rate significantly.

Push as a mechanism is also fundamentally less desirable than pull, since the control plane overhead consumed to load and maintain information about unused destinations is entirely wasted. The only potential downside to the pull option is the delay required for the demand-loading of information.

(It's also probably worth noting that many issues that some people have with the mapping approach of LISP, such as the total mapping database size, etc are the same - if not worse - for push as they are for pull.)

Finally, for IPv4, as the address space becomes more highly used, it will become more fragmented - i.e. there will tend to be more, smaller, entries. For a routing table, which every router has to hold, this is problematic. For a demand-loaded mapping table, it is not bad. Indeed, this was the original motivation for LISP ([RFC4984]) - although many other useful and desirable uses for it have since been enumerated (see [Introduction], Section "Applications").

For all of these reasons, as long as there is locality of reference (i.e. most ITRs will use only a subset of the entire set), it makes much more sense to use the a pull model, than the classic push one heretofore seen widely at the internetwork layer (with a pull approach thus being somewhat novel - and thus unsettling to many - to people who work at that layer).

It may well be that some sites (e.g. large content providers) may need non-standard mechanisms - perhaps something more of a 'push' model. This remains to be determined, but it is certainly feasible.

## 6.2. Caching of Mappings

It should be noted that the caching spoken of here is likely not classic caching, where there is a fixed/limited size cache, and entries have to be discarded to make room for newly needed entries. The economics of memory being what they are, there is no reason to discard mappings once they have been loaded (although of course implementations are free to choose to do so, if they wish to).

This leads to another point about the caching of mappings: the algorithms for management of the cache are purely a local issue. The algorithm in any particular ITR can be changed at will, with no need for any coordination. A change might be for purposes of experimentation, or for upgrade, or even because of environmental variations - different environments might call for different cache management strategies.

The local, unsynchronized replacability of the cache management scheme is the architectural aspect of the design; the exact algorithm, which is engineering, is not.

## 6.3. Amount of State

{{To be written.}} [Iannone]

- Mapping cache size
- Mention studies
- Delegation cache size (in MRs)
- Mention studies
- Any others?

#### 6.4. Scalability of The Indexing Subsystem

LISP initially used an indexing subsystem called ALT. [ALT] ALT was relatively easy to construct from existing tools (GRE, BGP, etc), but it had a number of issues that made it unsuitable for large-scale use. ALT is now being superseded by DDT. [DDT]

The basic structure and operation of DDT is identical to that of TREE, so the extensive simulation work done for TREE applies equally to DDT, as do the conclusions drawn about TREE's superiority to ALT. [Jakab]

From an architectural point of view, the main advantage of DDT is that it enables client side caching of information about intermediate nodes in the resolution hierarchy, and also enables direct communication with them. As a result, DDT has much better scaling properties than ALT.

The most important result of this change is that it avoids a concentration of resolution request traffic at the root of the indexing tree, a problem which by itself made ALT unsuitable for a global-scale system. The problem of root concentration (and thus overload) is almost unavoidable in ALT (even if masses of 'bypass' links are created).

ALT's scalability also depends on enforcing an intelligent organization that increases aggregation. Unfortunately, the current backbone routing BGP system shows that there is a risk of an organic growth of ALT, one which does not achieve aggregation. DDT does not display this weakness, since its organization is inherently hierarchical (and thus inherently aggregable).

The hierarchical organization of DDT also reduces the possibility for a configuration error which interferes with the operation of the network (unlike the situation with the current BGP DFZ). DDT security mechanisms can also help produce a high degree of robustness, both against misconfiguration, and deliberate attack. The direct communication with intermediate nodes in DDT also helps to quickly locate problems when they occur, resulting in better operational characteristics.

Next, since in ALT mapping requests must be transmitted through an overlay network, a significant share of requests can see substantially increased latencies. Simulation results in the TREE work clearly showed, and quantified, this effect.

The simulations also showed that the nodes composing the ALT and DDT networks for a mapping database of full Internet size could have thousands of neighbours. This is not an issue for DDT, but would almost certainly have been problematic for ALT nodes, since handling that number of simultaneous BGP sessions would likely to be difficult.

#### 7. Security

LISP does not yet have an overarching security architecture. Many parts of the system have been hardened, but more on a case-by case basis, rather than from an overall perspective. (This is in part due to the 'just enough' approach to security initially taken in LISP; see [Introduction], Section "Just Enough Security".)

This section represents an attempt to produce a more broadly-based view of security in LISP; it mostly resulted from an attempt to add security to the DDT indexing system ([DDT]), but the analysis is is general enough to apply to LISP broadly.

The \_good\_ thing about the Internet is that it brings the world to your doorstep - masses of information from all around the world are instantly available on your computing device. The \_bad\_ thing about the Internet is that it brings the world to your doorstep - including legions of crackers, thieves, and general scum and villainy. Thus, any node may be the target of fairly sophisticated attack - often automated (thereby reducing the effort required of the attacker to spread their attack as broadly as possible).

Security in LISP faces many of the same challenges as security for other parts of the Internet: good security usually work for the users, but without good security, things are vulnerable.

The Internet has seen many very secure systems devised, only to see them fail to reach wide adoption; the reasons for that are complex, and vary, but being too much work to use is a common thread. It is for this reason that LISP attempts to provide 'just enough' security (see [Introduction], Section "Just Enough Security").

### 7.1. Basic Philosophy

To square this circle, of needing to have very good security, but of it being too difficult to use very good security, the general concept is for LISP to have a series of 'graded' security measures available, with the 'ultimate' security mechanisms being very high-grade indeed.

The concept is to devise a plan in which LISP can simultaneously attempt to have not just 'ultimate' security, but also one or more 'easier' modes, ones which will be easier to configure and use. This 'easier' mode can be both an interim system (with the full powered system available for when it it needed), as well as the system used in sections of the network where security is less critical (following the general rule that the level of any security should generally be matched to what is being protected).

The challenge is to do this in a way that does not make the design more complex, since it has to include both the 'full strength' mechanism(s), and the 'easier to configure' mechanism(s). This is one of the fundamental tradeoffs to struggle with: it is easy to provide 'easier to configure' options, but that may make the overall design more complex.

As far as making it hard to implement to begin with (also something of a concern initially, although obviously not for the long term): we can make it 'easy' to deploy initially by simply not implementing/ configuring the heavy-duty security early on. (Provided, of course, that the packet formats, etc, needed to support such security are all included in the design to begin with.)

### 7.2. Design Guidance

In designing the security, there are a small number of key points that will guide the design:

- Design lifetime
- Threat level

How long is the design intended to last? If LISP is successful, a minimum of a 50-year lifetime is quite possible. (For comparison, IPv4 is now 34 at the time of writing this, and will be around for at least several decades yet, if not longer; DNS is 28, and will probably last indefinitely.)

How serious are the threats it needs to meet? As mentioned above, the Internet can bring the worst crackers from anywhere to any location, in a flash. Their sophistication level is rising all the time: as the easier holes are plugged, they go after others. This will inevitably eventually require the most powerful security mechanisms available to counteract their attacks.

Which is not to say that LISP needs to be that secure right away. The threat will develop and grow over a long time period. However, the basic design has to be capable of being securable to the expanded degree that will eventually be necessary. However, eventually it will need to be as securable as, say, DNS - i.e. it can be secured to the same level, although people may chose not to secure their LISP infrastructure as well as DNSSEC potentially does. [RFC4033]

In particular, it should be noted that historically many systems have been broken into, not through a weakness in the algorithms, etc, but because of poor operational mechanics. (The well-known 'Ultra' breakins of the Allies were mostly due to failures in operational procedure. [Welchman]) So operational capabilities intended to reduce the chance of human operational failure are just as important as strong algorithms; making things operationally robust is a key part of 'real' security.

#### 7.2.1. Security Mechanism Complexity

Complexity is bad for several reasons, and should always be reduced to a minimum. There are three kinds of complexity cost: protocol complexity, implementation complexity, and configuration complexity. We can further subdivide protocol complexity into packet format complexity, and algorithm complexity. (There is some overlap of algorithm complexity, and implementation complexity.)

We can, within some limits, trade off one kind of complexity for others: e.g. we can provide configuration options which are simpler for the users to operate, at the cost of making the protocol and implementation complexity greater. And we can make initial (less capable) implementations simpler if we make the protocols slightly more complex (so that early implementations don't have to implement all the features of the full-blown protocol).

It's more of a question of some operational convenience/etc issues - e.g. 'How easy will it be to recover from a cryptosystem compromise'. If we have two ways to recover from a security compromise, one which is mostly manual and a lot of work, and another which is more automated but makes the protocol more complicated, if compromises really are very rare, maybe the smart call is to go with the manual thing - as long as we have looked carefully at both options, and understood in some detail the costs and benefits of each.

#### 7.3. Security Overview

First, there are two different classes of attack to be considered: denial of service (DoS, i.e. the ability of an intruder to simply cause traffic not to successfully flow) versus exploitation (i.e. the ability to cause traffic to be 'highjacked', i.e. traffic to be sent to the wrong location).

Second, one needs to look at all the places that may be attacked. Again, LISP is a relatively simple system, so there are not that many parts to examine. The following are the things we need to secure:

- Lookups
- Indexing



- Mappings

#### 7.3.1. Securing Lookups

{{To be written.}} Nonces, [SecurityReq]

#### 7.3.2. Securing The Indexing Subsystem

It is envisioned that DDT will be highly securable, with all the delegations cryptographically secured via public-private signatures, very similar to the way DNS is ([RFC4033]).

The detailed mechanisms will be based on DNS's; this has the obvious benefit that all the lessons of DNS's years of practical experience with deployment, operations, etc, as well as the improvements to the basic design of DNS Security to provide a secure but usable system can be taken into account. However, DDT's security will also apply the thinking above, about making a 'versio' which is easier to use available.

{{To be written.}}

#### 7.3.3. Securing Mappings

There are two approaches to securing the provision of mappings. The first, which is of course not completely satisfactory, is to only secure the channel between the ITR and the entities involved in providing mappings for it. (See above, Section 7.3.1)

The second is to secure the mappings themselves, by signing them 'at birth' (much the same way in which DNS Security operates). [RFC4033]. There was an attempt early on to suggest such a system for LISP ([SecurityAuth]), but it was not adopted (although the particular proposal was rather complex).

In the long run, the latter approach would obviously be superior, since it would be almost immune to any compromises of the mapping distribution system. {{Tie-in to space allocation security}}

#### 7.4. Securing the xTRs

--- Cache management  
--- Unsolicited Map-Replies are \_very bad\_ - must go through mapping system to verify that the sender is authoritative for that range of EIDs

#### 8. Robustness

-- Depends on deployment as well as design  
-- Architected, visible replication of state/data  
-- Overlapping mechanisms (ref redundancy as key for robustness)

#### 9. Fault Discovery/Handling

Any global communication system must be robust, and to be robust, it must be able to discover and handle problems. LISP's general philosophy of robustness is usually to have overlapping, simple mechanisms to discover and repair problems.

#### 10. Optimization

-- Philosophy  
-- Piggybacking  
-- 'Wiretapping' return mappings  
--- Security is an issue on that

## 11. Open Issues

Although much work has been done on LISP, and it operates satisfactorily in a reasonably large initial deployment, there are a few potentially problematic issues which remain. It is not clear if they will be issues which need to be dealt, since they have not proven to be obstacles so far, but it is worth listing them.

We can divide them in \_local\_ issues, i.e. ones which can be solved on a node-by-node basis, without requiring co-ordinated change, and systemic issues, which are obviously more problematic, since they could require co-ordinated changes to the protocols.

### 11.1. Local Open Issues

#### 11.1.1. Missing Mapping Packet Queueing

Currently, some (all?) ITRs discard packets when they need a mapping, but have not loaded one yet, thereby causing the applicaton to have to retransmit their opening packet. True, many ARP implementations use the same strategy, but the average APR cache will only ever contain a few mappings, so it will not be so noticeable as with the mapping cache in an ITR, which will likely contain thousands.

Obviously, they could queue the packets while waiting to load the mapping, but this presents a number of subtle implementation issues: the ITR must make sure that it does not queue too many packets, etc.

In particular, if such packets are queued, this presents a potential DoS attack vector, unless the code is carefully written with that possibility in mind.

#### 11.1.2. Mapping Cache Management Algorithm

Relatively little work has been done on sophisticated mapping cache management algorithms; in particular, the issue of which mapping(s) to drop if the cache reaches some maximum allowed size.

This particular issue has also been identified as another potential DoS attack vector.

### 11.2. Systemic Open Issues

#### 11.2.1. Mapping Database Provider Lock-in

This refers to the fact that if one does not like the entity which is providing the indexing for the part of the address space which one's EIDs are allocated out of, there isn't probably isn't any way to switch to an alternative provider.

It is not clear that this is a real problem, though - the fact that all DNS top-level zones only have a single registry has not been a problem, nor has the fact that if one doesn't like the service the registry offers, one can't take one's DNS name to another registry.

Doing anything about it would also be difficult. Although it is \_technically\_ possible to duplicate any node in the delegation tree, and in theory such duplicates could be provided by different providers, it is not clear that such an arrangement would make \_business\_ sense.

For instance, if the holder of 10.1.1/24 decides they do not like the entity providing indexing for 10.1/16 (call them E1), and ask another entity (E2) to provide alternative service for 10.1/16, two problems arise. First, E1 is \_still\_ going to have to maintain the correct data for 10.1.1/24, and response to queries asking about them.

Second, E2 will similarly have to maintain data for, and reply to queries about, all the other space-holders in 10.1/16 - even though they will likely not have any business relationship with them.

#### 11.2.2. Automated ETR Synchronization

LISP requires that all the ETRs which are authoritative for the mappings for a particular address block return the same mapping data. In particular, their idea of the 'liveness' of all the ETRs should be identical, and correct.

At the moment, this is mostly a manual process, although liveness information can be currently be gathered from some IGPs.

#### 11.2.3. EID Reachability

At the moment, LISP assumes that if an ETR is reachable from a given ITR, all destination EIDs behind that ETR are reachable from that ETR. There is no way to detect if any are not, nor to switch to an alternate ETR.

It is not clear that this is a problem that needs attention. The same has been true for all border routers for many years now, and there does not seem to be any general mechanism to deal with it (Although some BGP implementations may advertize changes in reachability status if what they are seeing from their IGP changes.)

#### 11.2.4. Detect and Avoid Broken ETRs

{{To be written}}

### 12. Acknowledgments

The author would like thank all the members of the core LISP group for their willingness to allow him to add himself to their effort, and for their enthusiasm for whatever assistance he has been able to provide. He would also like to thank (in alphabetical order) Vina Ermagan, Vince Fuller, and Joel Halpern for their careful review of, and helpful suggestions for, this document. Grateful thanks also to Vince Fuller for help with XML.

A final thanks is due to John Wrocklawski for the author's organizational affiliation. This memo was created using the xml2rfc tool

### 13. IANA Considerations

This document makes no request of the IANA.

### 14. Security Considerations

This memo does not define any protocol and therefore creates no new security issues.

### 15. References

#### 15.1. Normative References

- |                |  |
|----------------|--|
| [DDT]          | V. Fuller, D. Lewis, and D. Farinacci, "LISP Delegated Database Tree", draft-fuller-lisp-ddt-01 (work in progress), March 2012.        |
| [Future]       | J. N. Chiappa, "Potential Long-Term Developments With the LISP System", draft-chiappa-lisp-evolution-00 (work in progress), July 2012. |
| [Introduction] | J. N. Chiappa, "An Introduction to the LISP Location-  |

Identity Separation System",  
draft-chiappa-lisp-introduction-00 (work in  
progress), July 2012.

- [SecurityAuth] R. Gagliano, "A Profile for Endpoint Identifier  
Origin Authorizations (IOA)",  
draft-rgaglian-lisp-iao-00 (work in progress),  
March 2009.
- [SecurityReq] F. Maino, V. Ermagan, A. Cabellos, D. Saucez, and  
O. Bonaventure, "LISP-Security (LISP-SEC)",  
draft-ietf-lisp-sec-02 (work in progress),  
March 2012.
- [AFI] IANA, "Address Family Indicators (AFIs)", Address  
Family Numbers, January 2011, <[http://www.iana.org/  
assignments/address-family-numbers](http://www.iana.org/assignments/address-family-numbers)>.

## 15.2. Informative References

- [RFC1631] K. Egevang and P. Francis, "The IP Network Address  
Translator (NAT)", RFC 1631, May 1994.
- [RFC1992] I. Castineyra, J. N. Chiappa, and M. Steenstrup, "The  
Nimrod Routing Architecture", RFC 1992, August 1996.
- [RFC3031] E. Rosen, A. Viswanathan, and R. Callon,  
"Multiprotocol Label Switching Architecture",  
RFC 3031, January 2001.
- [RFC4033] R. Arends, R. Austein, M. Larson, D. Massey, and  
S. Rose, "DNS Security: Introduction and  
Requirements", RFC 4033, March 2005.
- [RFC4423] R. Moskowitz and P. Nikander, "Host Identity Protocol  
(HIP) Architecture", RFC 4423, May 2006.
- [RFC4984] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB  
Workshop on Routing and Addressing", RFC 4984,  
September 2007.
- [RFC6115] T. Li, Ed., "Recommendation for a Routing  
Architecture", RFC 6115, February 2011.
- Perhaps the most ill-named RFC of all time; it  
contains nothing that could truly be called a  
'routing architecture'.
- [RFC6127] J. Arkko and M. Townsley, "IPv4 Run-Out and IPv4-IPv6  
Co-Existence Scenarios", RFC 6127, May 2011.
- [ALT] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis,  
"LISP Alternative Topology (LISP-ALT)",  
draft-ietf-lisp-alt-10 (work in progress),  
December 2011.
- [NERD] E. Lear, "NERD: A Not-so-novel EID to RLOC Database",  
draft-lear-lisp-nerd-09 (work in progress),  
April 2012.
- [ILNP] R.J. Atkinson and S.N. Bhatti, "ILNP Architectural  
Description", draft-irtf-rrg-ilnp-arch-05 (work in  
progress), May 2012.
- [Chiappa] J. N. Chiappa, "Endpoints and Endpoint Names: A  
Proposed Enhancement to the Internet Architecture",  
Personal draft (work in progress), 1999,

<<http://www.chiappa.net/~jnc/tech/endpoints.txt>>.

- [Jakab] L. Jakab, A. Cabellos-Aparicio, F. Coras, D. Saucez, and O. Bonaventure, "LISP-TREE: A DNS Hierarchy to Support the LISP Mapping System", in 'IEEE Journal on Selected Areas in Communications', Vol. 28, No. 8, pp. 1332-1343, October 2010.
- [Iannone] L. Iannone and O. Bonaventure, "On the Cost of Caching Locator/ID Mappings", in 'Proceedings of the 3rd International Conference on emerging Networking EXperiments and Technologies (CoNEXT'07)', ACM, pp. 1-12, December 2007.
- [McQuillan] J. M. McQuillan, W. R. Crowther, B. P. Cosell, D. C. Walden, and F. E. Heart, "Improvements in the Design and Performance of the ARPA Network", Proceedings AFIPS 1972 FJCC, Vol. 40, pp. 741-754.
- [Templin] F. Templin, "LISP WG", LISP WG list message, Message-ID: 39C363776A4E8C4A94691D2BD9D1C9A105B0AC71@XCH-NW-7V2.nw.nos.boeing.com, 13 March 2009,, <<http://www.ietf.org/mail-archive/web/lisp/current/msg00269.html>>.
- [Wasserman] M. Wasserman, "IPv6 networking: Bad news for small biz", IETF list message, Message-Id: D11C4A34-7362-423E-A60E-476FC5D61D37@lilacglade.org, 5 April 2012, <<https://www.ietf.org/ibin/c5i?mid=6&rid=49&gid=0&k1=933&k2=62733&tid=1340933524>>.
- [Welchman] G. Welchman, "The Hut Six Story", Allen Lane, London, pg. 3, 1982.
- A truly monumental book; the ground it covers ranges from his work helping break German codes in World War II to his experience with securing data packet networks!

## Appendix A. Glossary/Definition of Terms

- Address
- Locator
- EID
- RLOC
- ITR
- ETR
- xTR
- PITR
- PETR
- MR
- MS
- DFZ

## Appendix B. Other Appendices

- Location/Identity Separation Brief History
- LISP History
- Old models (LISP 1, LISP 1.5, etc)
- Different mapping distribution models (e.g. LISP-NERD)
- Different mapping indexing models (LISP-ALT forwarding/overlay model),  
LISP-TREE DNS-based, LISP-CONS)

Author's Address

J. Noel Chiappa  
Yorktown Museum of Asian Art  
Yorktown, Virginia  
USA

EMail: [jnc@mit.edu](mailto:jnc@mit.edu)

LISP Working Group  
Internet-Draft  
Intended status: Informational  
Expires: January 17, 2013

J. N. Chiappa  
Yorktown Museum of Asian Art  
July 16, 2012

An Introduction to the LISP Location-Identity Separation System  
draft-chiappa-lisp-introduction-01

Abstract

LISP is an upgrade to the architecture of the IPvN internetworking system, one which separates location and identity (currently intermingled in IPvN addresses). This is a change which has been identified by the IRTF as a critically necessary evolutionary architectural step for the Internet. In LISP, nodes have both a 'locator' (a name which says where in the network's connectivity structure the node is) and an 'identifier' (a name which serves only to provide a persistent handle for the node). A node may have more than one locator, or its locator may change over time (e.g. if the node is mobile), but it keeps the same identifier.

One of the chief novelties of LISP, compared to other proposals for the separation of location and identity, is its approach to deploying this upgrade. (In general, it is comparatively easy to conceive of new network designs, but much harder to devise approaches which will actually get deployed throughout the global network.) LISP aims to achieve the near-ubiquitous deployment necessary for maximum exploitation of an architectural upgrade by i) minimizing the amount of change needed (existing hosts and routers can operate unmodified); and ii) by providing significant benefits to early adopters.

This document is an introduction to the entire LISP system, for those who are unfamiliar with it. It is intended to be both easy to follow, and also give a fairly detailed understanding of the entire system.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to format it for publication as an RFC or to translate it into languages other than English.

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 <http://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 January 17, 2013.

Copyright Notice

Copyright (c) 2012 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 (<http://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. Background
2. Deployment Philosophy
  - 2.1. Economics
  - 2.2. Maximize Re-use of Existing Mechanism
  - 2.3. Self-Deployment
3. LISP Overview
  - 3.1. Basic Approach
  - 3.2. Basic Functionality
  - 3.3. Mapping from EIDs to RLOCs
  - 3.4. Interworking With Non-LISP-Capable Endpoints
4. Initial Applications
  - 4.1. Provider Independence
  - 4.2. Multi-Homing
  - 4.3. Traffic Engineering
  - 4.4. Mobility
  - 4.5. IP Version Reciprocal Traversal
  - 4.6. Local Uses
5. Major Functional Subsystems
  - 5.1. xTRs
  - 5.2. Mapping System
    - 5.2.1. Mapping System Organization
    - 5.2.2. Interface to the Mapping System
    - 5.2.3. Indexing Subsystem
6. Examples of Operation
  - 6.1. An Ordinary Packet's Processing
  - 6.2. A Mapping Cache Miss
7. Design Approach
  - 7.1. Quick Implement-Test Loop
    - 7.1.1. No Desk Fixes
    - 7.1.2. Code Before Documentation
  - 7.2. Only Fix Real Problems
  - 7.3. No Theoretical Perfection
    - 7.3.1. No Ocean Boiling
  - 7.4. Just Enough Security
8. xTRs
  - 8.1. When to Encapsulate
  - 8.2. UDP Encapsulation Details
  - 8.3. Header Control Channel
    - 8.3.1. Echo Nonces
    - 8.3.2. Instances
  - 8.4. Fragmentation
  - 8.5. Mapping Gleaning in ETRs
9. The Mapping System
  - 9.1. The Indexing Subsystem
  - 9.2. The Mapping System Interface
    - 9.2.1. Map-Request Messages
    - 9.2.2. Map-Reply Messages
    - 9.2.3. Map-Register and Map-Notify Messages
    - 9.2.4. Map-Referral Messages
  - 9.3. Reliability via Replication
  - 9.4. Extended Tools
  - 9.5. Expected Performance
10. Deployment Mechanisms
  - 10.1. Internetworking Mechanism
  - 10.2. Proxy Devices
    - 10.2.1. PITRs
    - 10.2.2. PETRs
  - 10.3. LISP-NAT



- 10.4. LISP and DFZ Routing
- 10.5. Use Through NAT Devices
  - 10.5.1. First-Phase NAT Support
  - 10.5.2. Second-Phase NAT Support
- 11. Current Improvements
  - 11.1. Mapping Versioning
  - 11.2. Replacement of ALT with DDT
    - 11.2.1. Why Not Use DNS
  - 11.3. Mobile Device Support
  - 11.4. Multicast Support
  - 11.5. {{Any others?}}
- 12. Fault Discovery/Handling
  - 12.1. Handling Missing Mappings
  - 12.2. Outdated Mappings
    - 12.2.1. Outdated Mappings - Updated Mapping
    - 12.2.2. Outdated Mappings - Wrong ETR
    - 12.2.3. Outdated Mappings - No Longer an ETR
  - 12.3. Erroneous mappings
  - 12.4. Neighbour Liveness
  - 12.5. Neighbour Reachability
- 13. Acknowledgments
- 14. IANA Considerations
- 15. Security Considerations
- 16. References
  - 16.1. Normative References
  - 16.2. Informative References
- Appendix A. Glossary/Definition of Terms
- Appendix B. Other Appendices

## 1. Background

It has gradually been realized in the networking community that networks (especially large networks) should deal quite separately with the identity and location of a node (basically, 'who' a node is, and 'where' it is). At the moment, in both IPv4 and IPv6, addresses indicate both where the named device is, as well as identify it for purposes of end-end communication.

The distinction was more than a little hazy at first: the early Internet [RFC791], like the ARPANET before it [Heart] [NIC8246], co-mingled the two, although there was recognition in the early Internet work that there were two different things going on. [IEN19]

This likely resulted not just from lack of insight, but also the fact that extra mechanism is needed to support this separation (and in the early days there were no resources to spare), as well as the lack of need for it in the smaller networks of the time. (It is a truism of system design that small systems can get away with doing two things with one mechanism, in a way that usually will not work when the system gets much larger.)

The ISO protocol architecture took steps in this direction [NSAP], but to the Internet community the necessity of a clear separation was definitively shown by Saltzer. [RFC1498] Later work expanded on Saltzer's, and tied his separation concepts into the fate-sharing concepts of Clark. [Clark], [Chiappa]

The separation of location and identity is a step which has recently been identified by the IRTF as a critically necessary evolutionary architectural step for the Internet. However, it has taken some time for this requirement to be generally accepted by the Internet engineering community at large, although it seems that this may finally be happening.

The LISP system for separation of location and identity resulted from the discussions of this topic at the Amsterdam IAB Routing and Addressing Workshop, which took place in October 2006. [RFC4984]

A small group of like-minded personnel from various scattered locations within Cisco, spontaneously formed immediately after that workshop, to work on an idea that came out of informal discussions at the workshop. The first Internet-Draft on LISP appeared in January, 2007, along with a LISP mailing list at the IETF. [LISP]

Trial implementations started at that time, with initial trial deployments underway since June 2007; the results of early experience have been fed back into the design in a continuous, ongoing process over several years. LISP at this point represents a moderately mature system, having undergone a long organic series of changes and updates.

LISP transitioned from an IRTF activity to an IETF WG in March 2009, and after numerous revisions, the basic specifications moved to becoming RFCs in 2012 (although work to expand and improve it continues, and undoubtedly will for a long time to come).

## 2. Deployment Philosophy

It may seem odd to cover 'deployment philosophy' at this point in such a document. However the deployment philosophy was a major driver for much of the design (to some degree the architecture, and to a very large measure, the engineering). So, as such an important motivator, it is very desirable for readers to have this material in hand as they examine the design, so that design choices that may seem questionable at first glance can be better understood.

Experience over the last several decades has shown that having a viable 'deployment model' for a new design is absolutely key to the success of that design. A new design may be fantastic - but if it can not or will not be successfully deployed (for whatever factors), it is useless. This absolute primacy of a viable deployment model is what has lead to some painful compromises in the design.

The extreme focus on a viable deployment scheme is one of the novelties of LISP.

### 2.1. Economics

The key factor in successful adoption, as shown by recent experience in the Internet - and little appreciated to begin with, some decades back - is economics: does the new design have benefits which outweigh its costs.

More importantly, this balance needs to hold for early adopters - because if they do not receive benefits to their adoption, the sphere of earliest adopters will not expand, and it will never get to widespread deployment. One might have the world's best clean-slate design, but if it does not have a deployment plan which is economically feasible, it's just a mildly interesting piece of paper.

This is particularly true of architectural enhancements, which are far less likely to be an addition which one can 'bolt onto the side' of existing mechanisms, and often offer their greatest benefits only when widely (or ubiquitously) deployed.

Maximizing the cost-benefit ratio obviously has two aspects. First, on the cost side, by making the design as inexpensive as possible, which means in part making the deployment as easy as possible. Second, on the benefit side, by providing many new capabilities, which is best done not by loading the design up with lots of features or options (which adds complexity), but by making the addition powerful through deeper flexibility. We believe LISP has met both of these goals.

## 2.2. Maximize Re-use of Existing Mechanism

One key part of reducing the cost of a new design is to absolutely minimize the amount of change `_required_` to existing, deployed, devices: the fewer devices need to be changed, and the smaller the change to those that do, the lower the pain (and thus the greater the likelihood) of deployment.

Designs which absolutely require 'forklift upgrades' to large amounts of existing gear are far less likely to succeed - because they have to have extremely large benefits to make their very substantial costs worthwhile.

It is for this reason that LISP, in most cases, initially requires no changes to devices in the Internet (both hosts and routers), and also initially reuses, wherever possible, existing protocols (IPv4 [RFC791] and IPv6 [RFC2460]). The 'initially' must be stressed - careful attention has also long been paid to the long-term future (see [Future]), and larger changes become feasible as deployment succeeds.

## 2.3. Self-Deployment

LISP has deliberately employed a rather different deployment model, which we might call 'self-deployment'; it does not require a huge push to get it deployed, rather, it is hoped that once people see it and realize they can easily make good use of it `_on their own_` (i.e. without requiring adoption by others), it will 'deploy itself' (hence the name of the approach).

One can liken the problem of deploying new systems in this way to rolling a snowball down a hill: unless one starts with a big enough initial snowball, and finds a hill of the right steepness (i.e. the right path for it to travel, once it starts moving), one's snowball is not going to go anywhere on its own. However, if one has picked one's spot correctly, little additional work is needed - just stand back and watch it go.

## 3. LISP Overview

LISP is an incrementally deployable architectural upgrade to the existing Internet infrastructure, one which provides separation of location and identity. The separation is usually not perfect, for reasons which are driven by the deployment philosophy (above), and explored in a little more detail elsewhere (in [Architecture], Section "Namespaces-EIDs-Residual").

LISP separates the functions of location and identity, current intermingled in IPvN addresses. (This document uses the meaning for 'address' proposed in [Atkinson], i.e. a name with mixed location and identity semantics.)

### 3.1. Basic Approach

In LISP, nodes have both a 'locator' (a name which says `_where_` in the network's connectivity structure the node is), called an 'RLOC', and an 'identifier' (a name which serves only to provide a persistent handle for the node), called an 'EID'. A node may have more than one RLOC, or its RLOC may change over time (e.g. if the node is mobile), but it keeps the same EID.

Technically, one should probably say that ideally, the EID names the node (or rather, its end-end communication stack, if one wants to be as forward-looking as possible), and the RLOC(s) name interface(s). (At the moment, in reality, the situation is somewhat more complex, as will be explained elsewhere (in [Architecture], Section "Namespaces-EIDs-Residual").

This second distinction, of `_what_` is named by the two classes of name, is necessary both to enable some of the capabilities that LISP provides (e.g the ability to seamlessly support multiple interfaces, to different networks), and is also a further enhancement to the architecture. Failing to clearly recognize both interfaces and communication stacks as distinctly separate classes of things is another failing of the existing Internet architecture (again, one inherited from the previous generation of networking).

A novelty in LISP is that it uses existing IPvN addresses (initially, at least) for both of these kinds of names, thereby minimizing the deployment cost, as well as providing the ability to easily interact with unmodified hosts and routers.

### 3.2. Basic Functionality

The basic operation of LISP, as it currently stands, is that LISP augmented packet switches near the source and destination of packets intercept traffic, and 'enhance' the packets.

The LISP device near the source looks up additional information about the destination, and then wraps the packet in an outer header, one which contains some of that additional information. The LISP device near the destination removes that header, leaving the original, unmodified, packet to be processed by the destination node.

The LISP device near the source (the Ingress Tunnel Router, or 'ITR') uses the information originally in the packet about the identity of its ultimate destination, i.e. the destination address, which one can view as the EID of the ultimate destination. It uses the destination EID to look up the current location (the RLOC) of that EID.

The lookup is performed through a 'mapping system', which is the heart of LISP: it is a distributed directory of bindings from EIDs to RLOCs. The destination RLOC will be (initially at least) the address of the LISP device near the destination (the Egress Tunnel Router, or 'ETR').

The ITR then generates a new outer header for the original packet, with that header containing the destination's RLOC as the wrapped packet's destination, and the ITR's own address (i.e. the RLOC of the original source) as the wrapped packet's source, and sends it off.

When the packet gets to the ETR, that outer header is stripped off, and the original packet is forwarded to the original ultimate destination for normal processing.

Return traffic is handled similarly, often (depending on the network's configuration) with the original ITR and ETR switching roles. The ETR and ITR functionality is usually co-located in a single device; these are normally denominated as 'xTRs'.

### 3.3. Mapping from EIDs to RLOCs

The mappings from EIDs to RLOCs are provided by a distributed (and potentially replicated) database, the mapping database, which is the heart of LISP.

Mappings are requested on need, not (generally) pre-loaded; in other words, mapping are distributed via a 'pull' mechanism. Once obtained by an ITR, they are cached, to limit the amount of control traffic to a practicable level. (The mapping system will be discussed in more detail below, in Section 5.2 and Section 9)

Extensive studies, including large-scale simulations driven by lengthy recordings of actual traffic at several major sites, have

been performed to verify that this 'pull and cache' approach is viable, in practical engineering terms. [Iannone] (This subject will be discussed in more detail in Section 9.5, below.)

### 3.4. Interworking With Non-LISP-Capable Endpoints

The capability for 'easy' interoperation between nodes using LISP, and existing non-LISP-using hosts or sites (often called 'legacy' hosts), is clearly crucial.

To allow such interoperation, a number of mechanisms have been designed. This multiplicity is in part because different mechanisms have different advantages and disadvantages (so that no single mechanism is optimal for all cases), but also because with limited field experience, it is not clear which (if any) approach will be preferable.

One approach uses proxy LISP devices, called PITRs (proxy ITRs) and PETRs (proxy ETRs), to provide LISP functionality during interaction with legacy sites. Another approach uses a device with combined LISP and NAT ([RFC1631]) functionality, named a LISP-NAT.

## 4. Initial Applications

As previously mentioned, it is felt that LISP will provide even the earliest adopters with some useful capabilities, and that these capabilities will drive early LISP deployment.

It is very important to note that even when used only for interoperation with existing unmodified hosts, use of LISP can still provide benefits for communications with the site which has deployed it - and, perhaps even more importantly, can do so to both sides. This characteristic acts to further enhance the utility for early adopters of deploying LISP, thereby increasing the cost/benefit ratio needed to drive deployment, and increasing the 'self-deployment' aspect of LISP.

Note also that this section only lists likely early applications and benefits - if and once deployment becomes more widespread, other aspects will come into play (as described in [Architecture], in the "Goals of LISP" section).

### 4.1. Provider Independence

Provider independence (i.e. the ability to easily change one's Internet Service Provider) was probably the first place where the Internet engineering community finally really felt the utility of separating location and identity.

The problem is simple: for the global routing to scale, addresses need to be aggregated (i.e. things which are close in the overall network's connectivity need to have closely related addresses), the so-called "provider aggregated" addresses. [RFC4116] However, if this principle is followed, it means that when an entity switches providers (i.e. it moves to a different 'place' in the network), it has to renumber, a painful undertaking. [RFC5887]

In theory, it ought to be possible to update the DNS entries, and have everyone switch to the new addresses, but in practice, addresses are embedded in many places, such as firewall configurations at other sites.

Having separate namespaces for location and identity greatly reduces the problems involved with renumbering; an organization which moves retains its EIDs (which are how most other parties refer to its nodes), but is allocated new RLOCs, and the mapping system can quickly provide the updated binding from the EIDs to the new RLOCs.

## 4.2. Multi-Homing

Multi-homing is another place where the value of separation of location and identity became apparent. There are several different sub-flavours of the multi-homing problem - e.g. depending on whether one wants open connections to keep working, etc - and other axes as well (e.g. site multi-homing versus host multi-homing).

In particular, for the 'keep open connections up' case, without separation of location and identity, the only currently feasible approach is to use provider-independent addresses - which moves the problem into the global routing system, with attendant costs. This approach is also not really feasible for host multi-homing.

Multi-homing was once somewhat esoteric, but a number of trends are driving an increased desirability, e.g. the wish to have multiple ISP links to a site for robustness; the desire to have mobile handsets connect up to multiple wireless systems; etc.

Again, separation of location and identity, and the existence of a binding layer which can be updated fairly quickly, as provided by LISP, is a very useful tool for all variants of this issue.

## 4.3. Traffic Engineering

Traffic engineering (TE) [RFC3272], desirable though this capability is in a global network, is currently somewhat problematic to provide in the Internet. The problem, fundamentally, is that this capability was not visualized when the Internet was designed, so support for it is somewhat in the 'when the only tool you have is a hammer, everything looks like nail' category.

TE is, fundamentally, a routing issue. However, the current Internet routing architecture, which is basically the Baran design of fifty years ago [Baran] (a single large, distributed computation), is ill-suited to provide TE. The Internet seems a long way from adopting a more-advanced routing architecture, although the basic concepts for such have been known for some time. [RFC1992]

Although the identity-location binding layer is thus a poor place, architecturally, to provide TE capabilities, it is still an improvement over the current routing tools available for this purpose (e.g. injection of more-specific routes into the global routing table). In addition, instead of the entire network incurring the costs (through the routing system overhead), when using a binding layer to provide TE, the overhead is limited to those who are actually communicating with that particular destination.

LISP includes a number of features in the mapping system to support TE. (Described in Section 5.2 below.)

## 4.4. Mobility

Mobility is yet another place where separation of location and identity is obviously a key part of a clean, efficient and high-functionality solution. Considerable experimentation has been completed on doing mobility with LISP.

## 4.5. IP Version Reciprocal Traversal

Note that LISP 'automagically' allows intermixing of various IP versions for packet carriage; IPv4 packets might well be carried in IPv6, or vice versa, depending on the network's configuration. This would allow an 'island' of operation of one type to be 'automatically' tunneled over a stretch of infrastructure which only supports the other type.

While the machinery of LISP may seem too heavyweight to be good for such a mundane use, this is not intended as a 'sole use' case for deployment of LISP. Rather, it is something which, if LISP is being deployed anyway (for its other advantages), is an added benefit that one gets 'for free'.

#### 4.6. Local Uses

LISP has a number of use cases which are within purely local contexts, i.e. not in the larger Internet. These fall into two categories: uses seen on the Internet (above), but here on a private (and usually small scale) setting; and applications which do not have a direct analog in the larger Internet, and which apply only to local deployments.

Among the former are multi-homing, IP version traversal, and support of VPN's for segmentation and multi-tenancy (i.e. a spatially separated private VPN whose components are joined together using the public Internet as a backbone).

Among the latter class, non-Internet applications which have no analog on the Internet, are the following example applications: virtual machine mobility in data centers; other non-IP EID types such as local network MAC addresses, or application specific data.

#### 5. Major Functional Subsystems

LISP has only two major functional subsystems - the collection of LISP packet switches (the xTRs), and the mapping system, which manages the mapping database. The purpose and operation of each is described at a high level below, and then, later on, in a fair amount of detail, in separate sections on each (Sections Section 8 and Section 9, respectively).

##### 5.1. xTRs

xTRs are fairly normal packet switches, enhanced with a little extra functionality in both the data and control planes, to perform LISP data and control functionality.

The data plane functions in ITRs include deciding which packets need to be given LISP processing (since packets to non-LISP sites may be sent 'vanilla'); looking up the mapping; encapsulating the packet; and sending it to the ETR. This encapsulation is done using UDP [RFC768] (for reasons to be explained below, in Section 8.2), along with an additional IPvN header (to hold the asource and destination RLOCs). To the extent that traffic engineering features are in use for a particular EID, the ITRs implement them as well.

In the ETR, the data plane simply unwraps the packets, and forwards the 'vanilla' packets to the ultimate destination.

Control plane functions in ITRs include: asking for {EID->RLOC} mappings via Map-Request control messages; handling the returning Map-Replies which contain the requested information; managing the local cache of mappings; checking for the reachability and liveness of their neighbour ETRs; and checking for outdated mappings and requesting updates.

In the ETR, control plane functions include participating in the neighbour reachability and liveness function (see Section 12.4); interacting with the mapping indexing system (next section); and answering requests for mappings (ditto).

##### 5.2. Mapping System

The mapping database is a distributed, and potentially replicated, database which holds bindings between EIDs (identity) and RLOCs (location). To be exact, it contains bindings between EID blocks and RLOCs (the block size is given explicitly, as part of the syntax).

Support for blocks is both for minimizing the administrative configuration overhead, as well as for operational efficiency; e.g. when a group of EIDs are behind a single xTR.

However, the block may be (and often is) as small as a single EID. Since mappings are only loaded upon demand, if smaller blocks become predominant, then the increased size of the overall database is far less problematic than if the routing table came to be dominated by such small entries.

A particular node may have more than one RLOC, or may change its RLOC(s), while keeping its singular identity.

The binding contains not just the RLOC(s), but also (for each RLOC for any given EID) priority and weight (to allow allocation of load between several RLOCs at a given priority); this allows a certain amount of traffic engineering to be accomplished with LISP.

#### 5.2.1. Mapping System Organization

The mapping system is actually split into two major functional subsystems. The actual bindings themselves are held by the ETRs, and an ITR which needs a binding effectively gets it from the ETR.

This co-location of the authoritative version of the mappings, and the forwarding functionality which it describes, is an instance of fate-sharing. [Clark]

To find the appropriate ETR(s) to query for the mapping, the second subsystem, an 'indexing system', itself also a distributed, potentially replicated database, provides information on which ETR(s) are authoritative sources of information about the bindings which are available.

#### 5.2.2. Interface to the Mapping System

The client interface to the mapping system from an ITR's point of view is not with the indexing system directly; rather, it is through devices called Map Resolvers (MRs).

ITRs send request control messages (Map-Request packets) to an MR. (This interface is probably the most important standardized interface in LISP - it is the key to the entire system.) The MR uses the indexing system to eventually forward the Map-Request to the appropriate ETR. The ETR formulates reply control messages (Map-Reply packets), which is conveyed to the ITR. The details of the indexing system, etc, are thus hidden from the 'ordinary' ITRs.

Similarly, the client interface to the indexing system from an ETR's point of view is through devices called Map Servers (MSs - admittedly a poorly chosen term, but it's too late to change it now).

ETRs send registration control messages (Map-Register packets) to an MS, which makes the information about the mappings which the ETR indicates it is authoritative for available to the indexing system. The MS formulates a reply control message (the Map-Notify packet), which confirms the registration, and is returned to the ETR. The details of the indexing system are thus likewise hidden from the 'ordinary' ETRs.

#### 5.2.3. Indexing Subsystem



The current indexing system is called the Delegated Database Tree (DDT), which is very similar in operation to DNS. [DDT], [RFC1034] However, unlike DNS, the actual mappings are not handled by DDT; DDT merely identifies the ETRs which hold the mappings.

Again, extensive large-scale simulations driven by lengthy recordings of actual traffic at several major sites, have been performed to verify the effectiveness of this particular indexing system. [Jakab]

## 6. Examples of Operation

To aid in comprehension, a few examples are given of user packets traversing the LISP system. The first shows the processing of a typical user packet, i.e. what the vast majority of user packets will see. The second shows what happens when the first packet to a previously-unseen destination (at a particular ITR) is to be processed by LISP.

### 6.1. An Ordinary Packet's Processing

This case follows the processing of a typical user packet (for instance, a normal TCP data or acknowledgment packet associated with an open HTTP connection) as it makes its way from the source host to the destination.

{{Rest to be written.}}

### 6.2. A Mapping Cache Miss

If a host sends a packet, and it gets to the ITR, and the ITR both i) determines that it needs to perform LISP processing on the user data packet, but ii) does not yet have a mapping cache entry which covers that destination EID, then more complex processing ensues.

{{Rest to be written.}}

## 7. Design Approach

Before describing LISP's components in more detail below, it may be worth saying a few words about the design philosophy used in creating them - this may make clearer the reasons for some engineering choices in the mechanisms given there.

### 7.1. Quick Implement-Test Loop

LISP uses a philosophy similar to that used in the early days of the Internet, which is to just build it, then try it and see what happens, and move forward from there based on what actually happens. The concept has been to get something up and running, and then modify it based on testing and experience.

#### 7.1.1. No Desk Fixes

Don't try and foresee all issues from desk analysis. (Which is not to say that one should not spend some time on trying to foresee problems, but be aware that it is a 'diminishing returns' process.) The performance of very large, complex, physically distributed systems is hard to predict, so rather than try (which would necessarily be an incomplete exercise anyway, testing would inevitably be required eventually), at a certain point it's better just to get on with it - and you will learn a host of other lessons in the process, too.

#### 7.1.2. Code Before Documentation

This is often a corollary to the kind of style described above. While it probably would not have been possible in a large,

inhomogenous group, the small, close nature of the LISP implementation group did allow this approach.

## 7.2. Only Fix Real Problems

Don't worry about anything unless experience show it's a real problem. For instance, in the early stages, much was made out of the problem of 'what does an ITR do if it gets a packet, but does not (yet) have a mapping for the destination?'

In practise, simply dropping such packets has just not proved to be a problem; the higher level protocol will retransmit them after a timeout, and the mapping is usually in place by then. So spending a lot of time (and its companion, energy) and mechanism (and its extremely undesirable companion, complexity) on solving this 'problem' would not have been the most efficient approach, overall.

## 7.3. No Theoretical Perfection

Attack hard problems with a number of cheap and simple mechanisms that co-operate and overlap. Trying to find a single mechanism that is all of:

- Robust
- Efficient
- Fast

is often (usually?) a fool's errand. (The analogy to the aphorism 'Fast, Cheap, Good - Pick Any Two' should be obvious.) However, a collection of simple and cheap mechanisms may effectively be able to meet all of these goals (see, for example, ETR Liveness/Reachability, Section 12.4).

Yes, this results in a system which is not provably correct in all circumstances. The world, however, is full of such systems - and in the real world, effective robustness is more likely to result from having multiple, overlapping mechanisms than one single high-powered (and inevitably complex) one. In the world of civil engineering, redundancy is now accepted as a key design principle; the same should be true of information systems. [Salvadori]

### 7.3.1. No Ocean Boiling

Don't boil the ocean to kill a single fish. This is a combination of 7.2 (Only Fix Real Problems) and 7.3 (No Theoretical Perfection); it just means that spending a lot of complexity and/or overhead to deal with a problem that's not really a problem is not good engineering.

## 7.4. Just Enough Security

How much security to have is a complex issue. It's relatively easy for designers to add good security, but much harder to get the users to jump over all the hoops necessary to use it. LISP has therefore adopted a position where we add 'just enough' security.

The overall approach to security in LISP is fairly subtle, though, and is covered in more detail elsewhere (in [Architecture], Section "Security").

## 8. xTRs

As mentioned above (in Section 5.1), xTRs are the basic data-handling devices in LISP. This section explores some advanced topics related to xTRs.

Careful rules have been specified for both TTL and ECN [RFC3168] to ensure that passage through xTRs does not interfere with the

operation of these mechanisms. In addition, care has been taken to ensure that 'traceroute' works when xTRs are involved.

### 8.1. When to Encapsulate

An ITR knows that a destination is running LISP, and thus that it should perform LISP processing on a packet (including potential encapsulation) if it has an entry in its local mapping cache that covers the destination EID.

Conversely, if the cache contains a 'negative' entry (indicating that the ITR has previously attempted to find a mapping that covers this EID, and it has been informed by the mapping system that no such mapping exists), it knows the destination is not running LISP, and the packet can be forwarded normally.

(The ITR cannot simply depend on the appearance, or non-appearance, of the destination in the DFZ routing tables, as a way to tell if a destination is a LISP site or not, because mechanisms to allow interoperation of LISP sites and 'legacy' sites necessarily involve advertising LISP sites' EIDs into the DFZ.)

### 8.2. UDP Encapsulation Details

The UDP encapsulation used by LISP for carrying traffic from ITR to ETR, and many of the details of how it works, were all chosen for very practical reasons.

Use of UDP (instead of, say, a LISP-specific protocol number) was driven by the fact that many devices filter out 'unknown' protocols, so adopting a non-UDP encapsulation would have made the initial deployment of LISP harder - and our goal (see Section 2.1) was to make the deployment as easy as possible.

The UDP source port in the encapsulated packet is a hash of the original source and destination; this is because many ISPs use multiple parallel paths (so-called 'Equal Cost Multi-Path'), and load-share across them. Using such a hash in the source-port in the outer header both allows LISP traffic to be load-shared, and also ensures that packets from individual connections are delivered in order (since most ISPs try to ensure that packets for a particular {source, source port, destination, destination port} tuple flow along a single path, and do not become disordered)..

The UDP checksum is zero because the inner packet usually already has a end-end checksum, and the outer checksum adds no value. [Saltzer] In most existing hardware, computing such a checksum (and checking it at the other end) would also present an intolerable load, for no benefit.

### 8.3. Header Control Channel

LISP provides a multiplexed channel in the encapsulation header. It is mostly (but not entirely) used for control purposes. (See [Architecture], Section "Architecture-Piggyback" for a longer discussion of the architectural implications of this.)

The general concept is that the header starts with an 8-bit 'flags' field, and it also includes two data fields (one 24 bits, one 32), the contents and meaning of which vary, depending on which flags are set. This allows these fields to be 'multiplexed' among a number of different low-duty-cycle functions, while minimizing the space overhead of the LISP encapsulation header.

#### 8.3.1. Echo Nonces

One important use is for a mechanism known as the Nonce Echo, which

is used as an efficient method for ITRs to check the reachability of correspondent ETRs.

Basically, an ITR which wishes to ensure that an ETR is up, and reachable, sends a nonce to that ETR, carried in the encapsulation header; when that ETR (acting as an ITR) sends some other user data packet back to the ITR (acting in turn as an ETR), that nonce is carried in the header of that packet, allowing the original ITR to confirm that its packets are reaching that ETR.

Note that lack of a response is not necessarily proof that something has gone wrong - but it strongly suggests that something has, so other actions (e.g. a switch to an alternative ETR, if one is listed; a direct probe; etc) are advised.

(See Section 12.5 for more about Echo Nonces.)

#### 8.3.2. Instances

Another use of these header fields is for 'Instances' - basically, support for VPN's across backbones. [RFC4026] Since there is only one destination UDP port used for carriage of user data packets, and the source port is used for multiplexing (above), there is no other way to differentiate among different destination address namespaces (which are often overlapped in VPNs).

#### 8.4. Fragmentation

Several mechanisms have been proposed for dealing with packets which are too large to transit the path from a particular ITR to a given ETR.

One, called the 'stateful' approach, keeps a per-ETR record of the maximum size allowed, and sends an ICMP Too Big message to the original source host when a packet which is too large is seen.

In the other, referred to as the 'stateless' approach, for IPv4 packets without the 'DF' bit set, too-large packets are fragmented, and then the fragments are forwarded; all other packets are discarded, and an ICMP Too Big message returned.

It is not clear at this point which approach is preferable.

#### 8.5. Mapping Gleaning in ETRs

As an optimization to the mapping acquisition process, ETRs are allowed to 'glean' mappings from incoming user data packets, and also from incoming Map-Request control messages. This is not secure, and so any such mapping must be 'verified' by sending a Map-Request to get an authoritative mapping. (See further discussion of the security implications of this in [Architecture], Section "Security-xTRs".)

The value of gleaning is that most communications are two-way, and so if host A is sending packets to host B (therefore needing B's EID->RLOC mapping), very likely B will soon be sending packets back to A (and thus needing A's EID->RLOC mapping). Without gleaning, this would sometimes result in a delay, and the dropping of the first return packet; this is felt to be very undesirable.

### 9. The Mapping System

RFC 1034 ("DNS Concepts and Facilities") has this to say about the DNS name to IP address mapping system:

"The sheer size of the database and frequency of updates suggest that it must be maintained in a distributed manner, with local

caching to improve performance. Approaches that attempt to collect a consistent copy of the entire database will become more and more expensive and difficult, and hence should be avoided."

and this observation applies equally to the LISP mapping system.

As previously mentioned, the mapping system is split into an indexing subsystem, which keeps track of where all the mappings are kept, and the mappings themselves, the authoritative copies of which are always held by ETRs.

### 9.1. The Indexing Subsystem

The indexing system in LISP is currently implemented by the DDT system. LISP initially used (for ease of getting something operational without having to write a lot of code) an indexing system called ALT, which used BGP running over virtual tunnels. [ALT] This proved to have a number of issues, and has now been superseded by DDT.

In DDT, the EID namespace(s) are instantiated as a tree of DDT nodes. Starting with the root node(s), which have 'responsibility' for the entire namespace, portions of the namespace are delegated to child nodes, in a recursive process extending through as many levels as are needed. Eventually, leaf nodes in the DDT tree delegate namespace blocks to ETRs.

MRs obtain information about delegations by interrogating DDT nodes, and caching the results. This allows them, when passed a request for a mapping by an ITR, to forward the mapping request to the appropriate ETR (perhaps after loading some missing delegation entries into their delegation cache).

### 9.2. The Mapping System Interface

As mentioned in Section 5.2.2, both of the interfaces to the mapping system (from ITRs, and ETRs) are standardized, so that the more numerous xTRs do not have to be modified when the mapping indexing system is changed. This precaution has already allowed the mapping system to be upgraded during LISP's evolution, when ALT was replaced by DDT.

This section describes the interfaces in a little more detail.

#### 9.2.1. Map-Request Messages

The Map-Request message contains a number of fields, the two most important of which are the requested EID block identifier (remember that individual mappings may cover a block of EIDs, not just a single EID), and the Address Family Identifier (AFI) for that EID block. [AFI] The inclusion of the AFI allows the mapping system interface (as embodied in these control packets) a great deal of flexibility. (See [Architecture], Section "Namespaces" for more on this.)

Other important fields are the source EID (and its AFI), and one or more RLOCs for the source EID, along with their AFIs. Multiple RLOCs are included to ensure that at least one is in a form which will allow the reply to be returned to the requesting ITR, and the source EID is used for a variety of functions, including 'gleaning' (see Section 8.5).

Finally, the message includes a long nonce, for simple, efficient protection against offpath attackers (see [Architecture], Section "Security-xTRs" for more), and a variety of other fields and control flag bits.

#### 9.2.2. Map-Reply Messages

The Map-Reply message looks similar, except it includes the mapping entry for the requested EID(s), which contains one or more RLOCs and their associated data. (Note that the reply may cover a larger block of the EID namespace than the request; most requests will be for a single EID, the one which prompted the query.)

For each RLOC in the entry, there is the RLOC, its AFI (of course), priority and weight fields (see Section 5.2), and multicast priority and weight fields.

#### 9.2.3. Map-Register and Map-Notify Messages

The Map-Register message contains authentication information, and a number of mapping records, each with an individual Time-To-Live (TTL). Each of the records contains an EID (potentially, a block of EIDs) and its AFI, a version number for this mapping (see Section 11.1), and a number of RLOCs and their AFIs.

Each RLOC entry also includes the same data as in the Map-Replies (i.e. priority and weight); this is because in some circumstances it is advantageous to allow the MS to proxy reply on the ETR's behalf to Map-Request messages. [Mobility]

Map-Notify messages have the exact same contents as Map-Register messages; they are purely acknowledgements.

#### 9.2.4. Map-Referral Messages

Map-Referral messages look almost identical to Map-Reply messages (which is felt to be an advantage by some people, although having a more generic record-based format would probably be better in the long run, as ample experience with DNS has shown), except that the RLOCs potentially name either i) other DDT nodes (children in the delegation tree), or ii) terminal MSs.

There are also optional authentication fields; see [Architecture], Section "Security-Mappings" for more.

#### 9.3. Reliability via Replication

Everywhere throughout the mapping system, robustness to operational failures is obtained by replicating data in multiple instances of any particular node (of whatever type). Map-Resolvers, Map-Servers, DDT nodes, ETRs - all of them can be replicated, and the protocol supports this replication.

There are generally no mechanisms specified yet to ensure coherence between multiple copies of any particular data item, etc - this is currently a manual responsibility. If and when LISP protocol adoption proceeds, an automated layer to perform this functionality can 'easily' be layered on top of the existing mechanisms.

#### 9.4. Extended Tools

In addition to the priority and weight data items in mappings, LISP offers other tools to enhance functionality, particularly in the traffic engineering area. One are 'source-specific mappings', i.e. the ETR may return different mappings to the enquiring ITR, depending on the identity of the ITR. This allows very fine-tuned traffic engineering, far more powerful than routing-based TE.

#### 9.5. Expected Performance

{{To be written.}}

#### 10. Deployment Mechanisms

This section discusses several deployment issues in more detail. With LISP's heavy emphasis on practicality, much work has gone into making sure it works well in the real-world environments most people have to deal with.

#### 10.1. Internetworking Mechanism

One aspect which has received a lot of attention are the mechanisms previously referred to (in Section 3.4) to allow interoperation of LISP sites with so-called 'legacy' sites which are not running LISP (yet).

To briefly refresh what was said there, there are two main approaches to such interworking: proxy nodes (PITRs and PETRs), and an alternative mechanism using device with combined NAT and LISP functionality; these are described in more detail here.

#### 10.2. Proxy Devices

PITRs (proxy ITRs) serve as ITRs for traffic from legacy hosts to nodes using LISP. PETRs (proxy ETRs) serve as ETRs for LISP traffic to legacy hosts (for cases where a LISP device cannot send packets directly to such sites, without encapsulation).

Note that return traffic to a legacy site from a LISP-using node does not necessarily have to pass through an ITR/PETR pair - the original packets can usually just be sent directly to the destination. However, for some kinds of LISP operation (e.g. mobile nodes), this is not possible; in these situations, the PETR is needed.

##### 10.2.1. PITRs

PITRs (proxy ITRs) serve as ITRs for traffic from legacy hosts to nodes using LISP. To do that, they have to advertise into the existing legacy backbone Internet routing the availability of whatever ranges of EIDs (i.e. of nodes using LISP) they are proxying for, so that legacy hosts will know where to send traffic to those LISP nodes.

As mentioned previously (Section 8.1), an ITR at another LISP site can avoid using a PITR (i.e. it can detect that a given destination is not a legacy site, if a PITR is advertising it into the DFZ) by checking to see if a LISP mapping exists for that destination.

This technique obviously has an impact on routing table in the DFZ, but it is not clear yet exactly what that impact will be; it is very dependent on the collected details of many individual deployment decisions.

A PITR may cover a group of EID blocks with a single EID advertisement, in order to reduce the number of routing table entries added. (In fact, at the moment, aggressive aggregation of EID announcements is performed, precisely to minimize the number of new announced routes added by this technique.)

At the same time, if a site does traffic engineering with LISP instead of fine-grained BGP announcement, that will help keep table sizes down (and this is true even in the early stages of LISP deployment). The same is true for multi-homing.

##### 10.2.2. PETRs

PETRs (proxy ETRs) serve as ETRs for LISP traffic to legacy hosts, for cases where a LISP device cannot send packets to sites without encapsulation. That typically happens for one of two reasons.

First, it will happen in places where some device is implementing Unicast Reverse Path Forwarding (uRPF), to prevent a variety of negative behaviour; originating packets with the source's EID in the source address field will result in them being filtered out and discarded.

Second, it will happen when a LISP site wishes to send packets to a non-LISP site, and the path in between does not support the particular IP protocol version used by the source along its entire length. Use of a PETR on the other side of the 'gap' will allow the LISP site's packet to 'hop over' the gap, by utilizing LISP's built-in support for mixed protocol encapsulation.

PETRs are generally paired with specific ITRs, which have the location of their PETRs configured into them. In other words, unlike normal ETRs, PETRs do not have to register themselves in the mapping database, on behalf of any legacy sites they serve.

Also, allowing an ITR to always send traffic leaving a site to a PETR does avoid having to choose whether or not to encapsulate packets; it can just always encapsulate packets, sending them to the PETR if it has no specific mapping for the destination. However, this is not advised: as mentioned, it is easy to tell if something is a legacy destination.

### 10.3. LISP-NAT

A LISP-NAT device, as previously mentioned, combines LISP and NAT functionality, in order to allow a LISP site which is internally using addresses which cannot be globally routed to communicate with non-LISP sites elsewhere in the Internet. (In other words, the technique used by the PITR approach simply cannot be used in this case.)

To do this, a LISP-NAT performs the usual NAT functionality, and translates a host's source address(es) in packets passing through it from an 'inner' value to an 'outer' value, and storing that translation in a table, which it can use to similarly process subsequent packets (both outgoing and incoming). [Interworking]

There are two main cases where this might apply:

- Sites using non-routable global addresses
- Sites using private addresses [RFC1918]

### 10.4. LISP and DFZ Routing

{{To be written.}}

### 10.5. Use Through NAT Devices

Like them or not (and NAT devices have many egregious issues - some inherent in the nature of the process of mapping addresses; others, such as the brittleness due to non-replicated critical state, caused by the way NATs were introduced, as stand-alone 'invisible' boxes), NATs are both ubiquitous, and here to stay for a long time to come.

Thus, in the actual Internet of today, having any new mechanisms function well in the presence of NATs (i.e. with LISP xTRs behind a NAT device) is absolutely necessary. LISP has produced a variety of mechanisms to do this.

#### 10.5.1. First-Phase NAT Support

The first mechanism used by LISP to operate through a NAT device only worked with some NATs, those which were configurable to allow inbound packet traffic to reach a configured host.



A pair of new LISP control messages, LISP Echo-Request and Echo-Reply, allowed the ETR to discover its temporary global address; the Echo-Request was sent to the configured Map-Server, and it replied with an Echo-Reply which included the source address from which the Echo Request was received (i.e. the public global address assigned to the ETR by the NAT). The ETR could then insert that address in any Map-Reply control messages which it sent to correspondent ITRs.

The fact that this mechanism did not support all NATs, and also required manual configuration of the NAT, meant that this was not a good solution; in addition, since LISP expects all incoming data traffic to be on a specific port, it was not possible to have multiple ETRs behind a single NAT (which normally would have only one global address to share, meaning port mapping would have to be used, except that... )

#### 10.5.2. Second-Phase NAT Support

For a more comprehensive approach to support of LISP xTR deployment behind NAT devices, a fairly extensive supplement to LISP, LISP NAT Traversal, has been designed. [NAT]

A new class of LISP device, the LISP Re-encapsulating Tunnel Router (RTR), passes traffic through the NAT, both to and from the xTR. (Inbound traffic has to go through the RTR as well, since otherwise multiple xTRs could not operate behind a single NAT, for the 'specified port' reason in the section above.)

(Had the Map-Reply included a port number, this could have been avoided - although of course it would be possible to define a new RLOC type which included protocol and port, to allow other encapsulation techniques.)

Two new LISP control messages (Info-Request and Info-Reply) allow an xTR to detect if it is behind a NAT device, and also discover the global IP address and UDP port assigned by the NAT to the xTR. A modification to LISP Map-Register control messages allows the xTR to initialize mapping state in the NAT, in order to use the RTR.

This mechanism addresses cases where the xTR is behind a NAT, but the xTR's associated MS is on the public side of the NAT; this limitation, that MS's must be in the 'public' part of the Internet, seems reasonable.

### 11. Current Improvements

In line with the philosophies laid out in Section 7, LISP is something of a moving target. This section discusses some of the contemporaneous improvements being made to LISP.

#### 11.1. Mapping Versioning

As mentioned, LISP has been under development for a considerable time. One early addition to LISP (it is already part of the base specification) is mapping versioning; i.e. the application of identifying sequence numbers to different versions of a mapping. [Versioning] This allows an ITR to easily discover when a cached mapping has been updated by a more recent variant.

Version numbers are available in control messages (Map-Replies), but the initial concept is that to limit control message overhead, the versioning mechanism should primarily use the multiplex user data header control channel (see Section 8.3).

Versioning can operate in both directions: an ITR can advise an ETR what version of a mapping it is currently using (so the ETR can

notify it if there is a more recent version), and ETRs can let ITRs know what the current mapping version is (so the ITRs can request an update, if their copy is outdated).

At the moment version numbers are manually assigned, and ordered. Some felt that this was non-optimal, and that a better approach would have been to have 'fingerprints' which were computed from the current mapping data (i.e. a hash). It is not clear that the ordering buys much (if anything), and the potential for mishaps with manually configured version numbers is self-evident.

## 11.2. Replacement of ALT with DDT

As mentioned in Section 9.2, an interface is provided to allow replacement of the indexing subsystem. LISP initially used an indexing system called ALT. [ALT] ALT was relatively easy to construct from existing tools (GRE, BGP, etc), but it had a number of issues that made it unsuitable for large-scale use. ALT is now being superseded by DDT.

As indicated previously (Section 9.5), the basic structure and operation of DDT is identical to that of TREE, so the extensive simulation work done for TREE applies equally to DDT, as do the conclusions drawn about TREE's superiority to ALT. [Jakab]

{{Briefly synopsise results}}

### 11.2.1. Why Not Use DNS

One obvious question is 'Since DDT is so similar to DNS, why not simply use DNS?' In particular, people are familiar with the DNS, how to configure it, etc - would it not thus be preferable to use it? To completely answer this would take more space than available here, but, briefly, there were two main reasons, and one lesser one.

First, the syntax of DNS names did not lend itself to looking up names in other syntaxes (e.g. bit fields). This is a problem which has been previously encountered, e.g. in reverse address lookups. [RFC5855]

Second, as an existing system, the interfaces between DNS (should it have been used as an indexing subsystem for LISP) would not be 'tuneable' to be optimal for LISP. For instance, if it were desired to have the leaf node in an indexing lookup directly contact the ETR on behalf of the node doing the lookup (thereby avoiding a round-trip delay), that would not be easy without modifications to the DNS code. Obviously, with a 'custom' system, this issue does not arise.

Finally, DNS security, while robust, is fairly complex. Doing DDT offered an opportunity to provide a more nuanced security model. (See [Architecture], Section "Security" for more about this.)

## 11.3. Mobile Device Support

Mobility is an obvious capability to provide with LISP. Doing so is relatively simple, if the mobile host is prepared to act as its own ETR. It obtains a local 'temporary use' address, and registers that address as its RLOC. Packets to the mobile host are sent to its temporary address, wherever that may be, and the mobile host first unwraps them (acting as an ETR), and then processes them normally (acting as a host).

(Doing mobility without having the mobile host act as its ETR is difficult, even if ETRs are quite common. The reason is that if the ETR and mobile host are not integrated, during the step from the ETR to the mobile host, the packets must contain the mobile host's EID, and this may not be workable. If there is a local router between the

ETR and mobile host, for instance, it is unlikely to know how to get the packets to the mobile host.)

If the mobile host migrates to a site which is itself a LISP site, things get a little more complicated. The 'temporary address' it gets is itself an EID, requiring mapping, and wrapping for transit across the rest of the Internet. A 'double encapsulation' is thus required at the other end; the packets are first encapsulated with the mobile node's temporary address as their RLOC, and then this has to be looked up in a second lookup cycle (see Section 8.1), and then wrapped again, with the site's RLOC as their destination.

This results in slight loss in maximum packet size, due to the duplicated headers, but on the whole it is considerably simpler than the alternative, which would be to re-wrap the packet at the site's ETR, when it is discovered that the destination's EID was not 'native' to the site. This would require that the mobile node's EID effectively have two different mappings, depending on whether the lookup was being performed outside the LISP site, or inside.

{{Also probably need to mention briefly how the other end is notified when mappings are updated, and about proxy-Map-Replies.}} [Mobility]

#### 11.4. Multicast Support

Multicast may seem an odd thing to support with LISP, since LISP is all about separating identity from location, but although a multicast group in some sense has an identity, it certainly does not have `_a_location`.

However, multicast is important to some users of the network, for a number of reasons: doing multiple unicast streams is inefficient; it is easy to use up all the upstream bandwidth, and without multicast a server can also be saturated fairly easily in doing the unicast replication. So it is important for LISP to 'play nicely' with multicast; work on multicast support in LISP is fairly advanced, although not far-ranging.

Briefly, destination group addresses are not mapped; only the source address (when the source is inside a LISP site) needs to be mapped, both during distribution tree setup, as well as actual traffic delivery. In other words, LISP's mapping capability is used: it is just applied to the source, not the destination (as with most LISP activity); the inner source is the EID, and the outer source is the EID's RLOC.

Note that this does mean that if the group is using separate source-specific trees for distribution, there isn't a separate distribution tree outside the LISP site for each different source of traffic to the group from inside the LISP site; they are all lumped together under a single source, the RLOC.

The approach currently used by LISP requires no packet format changes to existing multicast protocols. See [Multicast] for more; additional LISP multicast issues are discussed in [LISP], Section 12.

#### 11.5. {{Any others?}}

### 12. Fault Discovery/Handling

LISP is, in terms of its functionality, a fairly simple system: the list of failure modes is thus not extensive.

#### 12.1. Handling Missing Mappings

Handling of missing mappings is fairly simple: the ITR calls for the mapping, and in the meantime can either discard traffic to the

destination (as many ARP implementations do) [RFC826], or, if dropping the traffic is deemed undesirable, it can forward them via a 'default Pitr'.

A number of PitrS advertise all EID blocks into the backbone routing, so that any ITRs which are temporarily missing a mapping can forward the traffic to these default PitrS via normal transmission methods, where they are encapsulated and passed on.

## 12.2. Outdated Mappings

If a mapping changes once an ITR has retrieved it, that may result in traffic to the EIDs covered by that mapping failing. There are three cases to consider:

- When the ETR traffic is being sent to is still a valid ETR for that EID, but the mapping has been updated (e.g. to change the priority of various ETRs)
- When the ETR traffic is being sent to is still an ETR, but no longer a valid ETR for that EID
- When the ETR traffic is being sent to is no longer an ETR

### 12.2.1. Outdated Mappings - Updated Mapping

A 'mapping versioning' system, whereby mappings have version numbers, and ITRs are notified when their mapping is out of date, has been added to detect this, and the ITR responds by refreshing the mapping. [Versioning]

### 12.2.2. Outdated Mappings - Wrong ETR

{{To be written.}}

### 12.2.3. Outdated Mappings - No Longer an ETR

If the destination of traffic from an ITR is no longer an ETR, one might get an ICMP Destination Unreachable error message. However, one cannot depend on that. The following mechanism will work, though.

Since the destination is not an ETR, the echoing reachability detection mechanism (see Section 8.3.1) will detect a problem. At that point, the backstop mechanism, Probing, will kick in. Since the destination is still not an ETR, that will fail, too.

At that point, traffic will be switched to a different ETR, or, if none are available, a re-map may be requested.

## 12.3. Erroneous mappings

{{To be written.}}

## 12.4. Neighbour Liveness

The ITR, like all packet switches, needs to detect, and react, when its next-hop neighbour ceases operation. As LISP traffic is effectively always unidirectional (from ITR to ETR), this could be somewhat problematic.

Solving a related problem, neighbour reachability (below) subsumes handling this fault mode, however.

Note that the two terms (liveness and reachability) are not synonymous (although a lot of LISP documentation confuses them). Liveness is a property of a node - it is either up and functioning, or it is not. Reachability is only a property of a particular pair of nodes.

If packets sent from a first node to a second are successfully received at the second, it is 'reachable' from the first. However, the second node may at the very same time not be reachable from some other node. Reachability is always a ordered pairwise property, and of a specified ordered pair.

#### 12.5. Neighbour Reachability

A more significant issue than whether a particular ETR E is up or not is, as mentioned above, that although ETR E may be up, attached to the network, etc, an issue in the network between a source ITR I and E may prevent traffic from I from getting to E. (Perhaps a routing problem, or perhaps some sort of access control setting.)

The one-way nature of LISP traffic makes this situation hard to detect in a way which is economic, robust and fast. Two out of the three are usually not too hard, but all three at the same time - as is highly desirable for this particular issue - are harder.

In line with the LISP design philosophy (Section 7.3), this problem is attacked not with a single mechanism (which would have a hard time meeting all those three goals simultaneously), but with a collection of simpler, cheaper mechanisms, which collectively will usually meet all three.

They are reliance on the underlying routing system (which can of course only reliably provide a negative reachability indication, not a positive one), the echo nonce (which depends on some return traffic from the destination xTR back to the source), and finally direct 'pinging', in the case where no positive echo is returned.

(The last is not the first choice, as due to the large fan-out expected of LISP devices, reliance on it as a sole mechanism would produce a fair amount of overhead.)

#### 13. Acknowledgments

The author would like thank all the members of the core LISP group for their willingness to allow him to add himself to their effort, and for their enthusiasm for whatever assistance he has been able to provide. He would also like to thank (in alphabetical order) Vina Ermagan, Vince Fuller, and especially Joel Halpern for their careful review of, and helpful suggestions for, this document. Grateful thanks also to Darrel Lewis for his help with material on non-Internet uses of LISP, and to Vince Fuller for help with XML.

A final thanks is due to John Wrocklawski for the author's organizational affiliation. This memo was created using the xml2rfc tool

#### 14. IANA Considerations

This document makes no request of the IANA.

#### 15. Security Considerations

This memo does not define any protocol and therefore creates no new security issues.

#### 16. References

##### 16.1. Normative References

[RFC768] J. Postel, "User Datagram Protocol", RFC 768, August 1980.

- [RFC791] J. Postel, "Internet Protocol", RFC 791, September 1981.
- [RFC1498] J. H. Saltzer, "On the Naming and Binding of Network Destinations", RFC 1498, (Originally published in: "Local Computer Networks", edited by P. Ravasio et al., North-Holland Publishing Company, Amsterdam, 1982, pp. 311-317.), August 1993.
- [RFC2460] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [Architecture] J.N. Chiappa, "The Architecture of the LISP Location-Identity Separation System", draft-chiappa-lisp-architecture-00 (work in progress), July 2012.
- [DDT] V. Fuller, D. Lewis, and D. Farinacci, "LISP Delegated Database Tree", draft-fuller-lisp-ddt-01 (work in progress), March 2012.
- [Future] J. N. Chiappa, "Potential Long-Term Developments With the LISP System", draft-chiappa-lisp-evolution-00 (work in progress), July 2012.
- [Interworking] D. Lewis, D. Meyer, D. Farinacci, and V. Fuller, "Interworking LISP with IPv4 and IPv6", draft-ietf-lisp-interworking-06 (work in progress), March 2012.
- [LISP] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-23 (work in progress), May 2012.
- [Mobility] D. Farinacci, V. Fuller, D. Lewis, and D. Meyer, "LISP Mobility Architecture", draft-meyer-lisp-mn-07 (work in progress), April 2012.
- [Multicast] D. Farinacci, D. Meyer, J. Zwiebel, and S. Venaas, "LISP for Multicast Environments", draft-ietf-lisp-multicast-14 (work in progress), February 2012.
- [NAT] V. Ermagan, D. Farinacci, D. Lewis, J. Skriver, F. Maino, and C. White, "NAT traversal for LISP", draft-ermagan-lisp-nat-traversal-01 (work in progress), March 2012.
- [Versioning] L. Iannone, D. Saucez, and O. Bonaventure, "LISP Mapping Versioning", draft-ietf-lisp-map-versioning-09 (work in progress), March 2012.
- [AFI] IANA, "Address Family Indicators (AFIs)", Address Family Numbers, January 2011, <<http://www.iana.org/assignments/address-family-numbers>>.

## 16.2. Informative References

- [NIC8246] A. McKenzie and J. Postel, "Host-to-Host Protocol for the ARPANET", NIC 8246, Network Information Center, SRI International, Menlo Park, CA, October 1977.
- [IEN19] J. F. Shoch, "Inter-Network Naming, Addressing, and Routing", IEN (Internet Experiment Note) 19, January 1978.

- [RFC826] D. Plummer, "Ethernet Address Resolution Protocol", RFC 826, November 1982.
- [RFC1034] P. V. Mockapetris, "Domain Names - Concepts and Facilities", RFC 1034, November 1987.
- [RFC1631] K. Egevang and P. Francis, "The IP Network Address Translator (NAT)", RFC 1631, May 1994.
- [RFC1918] Y. Rekhter, R. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, "Address Allocation for Private Internets", RFC 1918, February 1996.
- [RFC1992] I. Castineyra, J. N. Chiappa, and M. Steenstrup, "The Nimrod Routing Architecture", RFC 1992, August 1996.
- [RFC3168] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.
- [RFC3272] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "Overview and Principles of Internet Traffic Engineering", RFC 3272, May 2002.
- [RFC4026] L. Andersson and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, March 2005.
- [RFC4116] J. Abley, K. Lindqvist, E. Davies, B. Black, and V. Gill, "IPv4 Multihoming Practices and Limitations", RFC 4116, July 2005.
- [RFC4984] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB Workshop on Routing and Addressing", RFC 4984, September 2007.
- [RFC5855] J. Abley and T. Manderson, "Nameservers for IPv4 and IPv6 Reverse Zones", RFC 5855, May 2010.
- [RFC5887] B. Carpenter, R. Atkinson, and H. Flinck, "Renumbering Still Needs Work", RFC 5887, May 2010.
- [ALT] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "LISP Alternative Topology (LISP-ALT)", draft-ietf-lisp-alt-10 (work in progress), December 2011.
- [NSAP] International Organization for Standardization, "Information Processing Systems - Open Systems Interconnection - Basic Reference Model", ISO Standard 7489.1984, 1984.
- [Atkinson] R. Atkinson, "Revised draft proposed definitions", RRG list message, Message-Id: 808E6500-97B4-4107-8A2F-36BC913BE196@extremenetworks.com, 11 June 2007, <<http://www.ietf.org/mail-archive/web/ram/current/msg01470.html>>.
- [Baran] P. Baran, "On Distributed Communications Networks", IEEE Transactions on Communications Systems Vol. CS-12 No. 1, pp. 1-9, March 1964.
- [Chiappa] J. N. Chiappa, "Endpoints and Endpoint Names: A Proposed Enhancement to the Internet Architecture", Personal draft (work in progress), 1999, <<http://www.chiappa.net/~jnc/tech/endpoints.txt>>.

- [Clark] D. D. Clark, "The Design Philosophy of the DARPA Internet Protocols", in 'Proceedings of the Symposium on Communications Architectures and Protocols SIGCOMM '88', pp. 106-114, 1988.
- [Heart] F. E. Heart, R. E. Kahn, S. M. Ornstein, W. R. Crowther, and D. C. Walden, "The Interface Message Processor for the ARPA Computer Network", Proceedings AFIPS 1970 SJCC, Vol. 36, pp. 551-567.
- [Jakab] L. Jakab, A. Cabellos-Aparicio, F. Coras, D. Saucez, and O. Bonaventure, "LISP-TREE: A DNS Hierarchy to Support the LISP Mapping System", in 'IEEE Journal on Selected Areas in Communications', Vol. 28, No. 8, pp. 1332-1343, October 2010.
- [Iannone] L. Iannone and O. Bonaventure, "On the Cost of Caching Locator/ID Mappings", in 'Proceedings of the 3rd International Conference on emerging Networking EXperiments and Technologies (CoNEXT'07)', ACM, pp. 1-12, December 2007.
- [Saltzer] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-To-End Arguments in System Design", ACM TOCS, Vol 2, No. 4, pp 277-288, November 1984.
- [Salvadori] M. Salvadori and M. Levy, "Why Buildings Fall Down", W. W. Norton, New York, pg. 81, 1992.

#### Appendix A. Glossary/Definition of Terms

- Address
- Locator
- EID
- RLOC
- ITR
- ETR
- xTR
- PITR
- PETR
- MR
- MS
- DFZ

#### Appendix B. Other Appendices

Possible appendices:

- Location/Identity Separation Brief History
- LISP History
- Old models (LISP 1, LISP 1.5, etc)

#### Author's Address

J. Noel Chiappa  
 Yorktown Museum of Asian Art  
 Yorktown, Virginia  
 USA

EMail: jnc@mit.edu



Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: January 10, 2013

F. Coras  
A. Cabellos-Aparicio  
J. Domingo-Pascual  
Technical University of  
Catalonia  
F. Maino  
D. Farinacci  
cisco Systems  
July 9, 2012

LISP Replication Engineering  
draft-coras-lisp-re-00

Abstract

This document describes a method to build and optimize inter-domain LISP router distribution trees for locator-based unicast and multicast replication of EID-based multicast packets.

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 <http://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 January 10, 2013.

Copyright Notice

Copyright (c) 2012 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 (<http://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 . . . . .	4
3. Overview . . . . .	5
4. Overlay Distribution Tree . . . . .	6
4.1. LISP Replication Node Database . . . . .	7
4.2. Building the Distribution Tree . . . . .	7
5. Distribution Tree Optimization . . . . .	8
5.1. Topology Discovery . . . . .	9
5.2. Optimization Algorithm . . . . .	9
6. Security Considerations . . . . .	11
7. IANA Considerations . . . . .	11
8. Acknowledgements . . . . .	11
9. References . . . . .	11
9.1. Normative References . . . . .	11
9.2. Informative References . . . . .	12
Appendix A. MADDBST heuristic . . . . .	13
Authors' Addresses . . . . .	13

## 1. Introduction

The Locator/Identifier Separation Protocol (LISP) [I-D.ietf-lisp] provides the mechanisms for the separation of Location and Identity semantics presently overloaded by IP addresses. The split results in the creation of two namespaces that unambiguously identify edge-site network objects, Endpoint IDs (EIDs), and core routing objects, Routing LOCators (RLOCs). Apart from aiding the scalability of the core routing infrastructure, the decoupling also enables the (re)implementation of new or existing inter-domain routing mechanisms.

One such mechanism is inter-domain IP source-specific multicast (SSM) [RFC4607]. In this sense, [I-D.ietf-lisp-multicast] defines the procedures carried out for delivering multicast packets from a source host in a LISP site to receivers residing in the same domain or in other LISP or non-LISP sites when an underlying multicast infrastructure exists. The signaling protocol it specifies for conveying (S-EID,G) state and building the distribution tree connecting the xTRs of the source and receiver domains is PIM [RFC4601]. An alternative method that uses Map-Requests instead of PIM for propagating (S-EID,G) state from multicast receiver site ETRs to source site ITR is established in [I-D.farinacci-lisp-mr-signaling].

Although desirable to use multicast routing in the core network when available, a mismatch between the multicast capabilities of receiver ETRs and source ITR might impede their multicast interconnection. In such a case, unicast RLOC encapsulation will be necessary to deliver multicast packets directly to the ETRs. This however leads to high ITR head-end replication for large sets of receiving ETRs. Therefore, to reduce the replication load of the ITR and scale the service with the number of multicast receivers, the ITR may choose to offload replication to a set of RTRs.

The current document describes how multicast RTRs can be used to build an inter-domain distribution tree rooted at the ITR that can perform unicast and/or multicast encapsulated replication of multicast packets. This concept, of distributing the replication load from ITR to other RTRs downstream on the core overlay distribution tree, is known as Replication Engineering or LISP-RE. Since unicast replication in such overlays can be suboptimal when compared to the underlay network, methods to optimize packet delivery over the distribution tree are also presented.

This specification does not describe how (S-EID,G) state is built in source and receiver domains, nor does it describe how such state is propagated from receiver ETRs to source ITR. This specification

defines how (S-EID,G) map-cache state is built in the ITR, RTRs and ETRs participating in the overlay distribution tree when they are not connectable by multicast.

## 2. Definition of Terms

The terminology in this document is consistent with the definitions in [I-D.ietf-lisp] and [I-D.ietf-lisp-multicast] however, it is extended to account for LISP-RE concepts:

**Delivery Group (DG):** The outer destination address of a multicast encapsulated multicast packet with an EID source.

**Replication Target:** A unicast locator or Delivery Group towards which a multicast packet may be encapsulated and replicated.

**Replication List:** A locator-set associated to a multicast map-cache entry (S-EID,G) as defined in [I-D.farinacci-lisp-te]. Locators in the set may be either unicast RLOCs or Delivery Groups. When used by a LISP router, a multicast packet matching the map-cache entry must be replicated to all members of the set. The unicast RLOCs are used to encapsulate multicast packets for unicast delivery on the underlying network. Delivery Groups are used to encapsulate multicast packets for multicast delivery on the underlay.

**Unicast Replication:** Is the notion of replicating a multicast packet with an EID source address at an ITR or RTR by encapsulating it into a unicast packet. That is, the oif-list of a multicast map-cache entry can not only have interfaces present for link-layer replication and multicast encapsulation but also for unicast encapsulation.

**Overlay Distribution Tree:** A degree-constrained spanning tree that represents the path followed by unicast and/or multicast encapsulated multicast packets from the root (ITR) to the leaves (ETRs) through intermediary nodes (RTRs). The ITR and RTRs unicast and/or multicast replicate packets to their tree children. Such tree is built and maintained by the overlay distribution tree controller either by using LISP signaling defined in [I-D.ietf-lisp-multicast] and [I-D.farinacci-lisp-mr-signaling] or by programming the mapping database directly by using ELPs to describe network-wide fanout.

Distribution Tree Controller (DTC): A central entity that manages the overlay distribution tree, such entity can be either the ITR or an external orchestration system.

LISP Replication Node: A router (either the ITR or an RTR) participating and replicating packets downstream in the distribution tree.

Multicast Ingress Tunnel Router (ITR): An ITR as specified in [I-D.ietf-lisp] that participates as the root in the distribution tree.

Multicast Egress Tunnel Router (ETR): An ETR as specified in [I-D.ietf-lisp] that participates as a leaf in the distribution tree. ETR are the only members of the tree that do not unicast replicate.

Multicast Re-encapsulating Tunnel Router (RTR): An RTR as specified in [I-D.farinacci-lisp-te] that participates as an intermediary node in the distribution tree.

Explicit Locator Path (ELP): an explicit and strictly ordered list of replication targets a packet must travel to. An ELP may be used to source route a LISP-encapsulated packet on an explicit path of RTRs, however the path between two RTRs is determined by the underlying routing protocol. ELP format is described in [I-D.farinacci-lisp-lcaf] and their use in [I-D.farinacci-lisp-te].

### 3. Overview

This specification describes a method to diminish the replication load of the ITR by using RTRs to build an inter-domain distribution tree. The tree is centrally managed either by the ITR itself or by an external orchestration system. An advantage of this orchestration system is that it offloads signaling from the ITR. The entity that manages the tree is generally referred to as the distribution tree controller (DTC).

In order to offload unicast replication of multicast packets the DTC uses a ITR and a set of RTRs. RTRs willing to participate in the distribution tree associated to the (S-EID,G) multicast channel must join the distribution tree by sending a Map-Request/Join-Request [I-D.farinacci-lisp-mr-signaling] to the DTC. Using this procedure the DTC learns the RLOCs of the available RTRs. Additionally, the DTC must learn the replication capacity of each RTR using out-of-band signaling or by manual configuration.

Given that the ITR and RTRs have a limited replication capacity the distribution tree is a degree-constrained spanning-tree. This means that the root is the ITR, the intermediary members are RTRs while leaves are always ETRs. Multicast packets are addressed to (S-EID,G) and are unicast and/or multicast encapsulated when being transported downstream the tree.

In order to build and maintain the overlay distribution tree the DTC must configure state in the replication nodes (ITR and RTRs). This is done by means of the signaling specified in [I-D.ietf-lisp-multicast] and [I-D.farinacci-lisp-mr-signaling]. Particularly, the DTC receives Map-Requests from RTRs (also from the ITR if the DTC is an external orchestration system) addressed to (S-EID,G). Upon inspection of the source RLOC of the Map-Request the controller determines the originating ITR/RTR and generates an ad-hoc Map-Reply containing the specific replication list for that particular node according to the topology of the tree. For a LISP replication node, the replication list specifies the set of RTRs/ETRs to which it has to replicate packets, i.e., its overlay distribution tree children. Alternatively, an external orchestration system may directly program the mapping database with ELPs that describe the topology of the overlay distribution tree. Ways of achieving this will be discussed in future versions of the document.

The DTC determines the specific topology of the overlay distribution tree using a centralized algorithm. The only requirements for this algorithm are that it builds a tree that guarantees that ETRs receive the encapsulated multicast packets, that the replication capacity of the ITR and RTRs is not exceeded and that forwarding loops are avoided.

In some cases the network administrator may want an optimized overlay distribution tree, although this specification does not standardize any particular algorithm it suggests one in Section 5.2. In order to build an optimized tree this algorithm makes use of the distance (e.g., latency) between the tree members and the amount of multicast receivers connected to each ETR. Such metrics are not provided by LISP and therefore must be obtained using out-of-band signaling.

#### 4. Overlay Distribution Tree

This section describes how the DTC can build an overlay distribution tree using the signaling and mechanisms defined in [I-D.ietf-lisp-multicast] and [I-D.farinacci-lisp-mr-signaling].

#### 4.1. LISP Replication Node Database

The DTC maintains per (S-EID,G) multicast channel a LISP Replication Node Database (LRND) that stores information about the distribution tree state. This information includes among others the RLOCs of the ITR, RTRs and ETRs that constitute the distribution tree and define the overlay replication topology (i.e., the parent-child relations). Said data may be obtained by the DTC from the standard signaling messages exchanged with the RTRs and ETRs. Additionally, by means of out-of-band signalling the DTC should obtain information about the replication capacity of RTRs.

If the operator chooses to build an optimized tree, more information must be available at the LRND, this is further discussed in Section 5.2.

#### 4.2. Building the Distribution Tree

This section describes the procedures followed by ETRs and RTRs when attaching to the distribution tree. All procedures assume that the DTC has a LRND consistent with the state of the overlay distribution tree and is aware of the replication capacity of participating RTRs.

The decision of an RTR to join the overlay distribution tree depends on out-of-band signalling (e.g., orchestration system, manual configuration). But, its attachment to the distribution tree is done by means of one of the following two procedures:

1. The RTR explicitly signals the ITR by sending a Join-Request for (S-EID,G) and is replied to with a replication list.
2. If an orchestration system programs the mapping database with ELPs describing the overlay distribution tree, an RTR Map-Requests for (S-EID,G) and receives as reply an ELP that defines its distribution tree fanout. Ways of encoding the tree topology into ELPs will be discussed in future versions of this document.

For RTRs using option 1 the DTC, an ITR in this case, will perform the same processing as for joining ETRs. The following sequence of steps is used to attach an ETR to the overlay distribution tree:

1. The DTC receives a Map-Request/Join-Request for (S-EID,G) from an ETR.
2. If multicast replication is requested, the DTC proceeds as defined in [I-D.farinacci-lisp-mr-signaling] and no further steps are taken.

3. If unicast replication is requested, the DTC must choose a position for the ETR in the distribution tree topology. Specifically, it initiates a search within the LRND for a node (either the ITR or a RTR) with enough spare replication capacity that will replicate multicast traffic towards the ETR. This tree member will become the parent of the ETR and once it is selected the LRND is updated accordingly. The search algorithm depends on operational requirements and this specification does not standardize one, however Section 5.2 provides an example algorithm. Note also that certain algorithms may require the complete or partial re-shape the tree based on certain performance metrics.
4. The DTC must create/update the (S-EID,G) associated replication state for the selected parent using the mechanisms defined in [I-D.ietf-lisp] and [I-D.farinacci-lisp-mr-signaling] (e.g., Solicit-Map-Request). This results in the parent sending a Map-Request for (S-EID,G), in turn, the DTC Map-Replies with an ad-hoc replication list of locator-sets according to topology stored at the LRND. If the algorithm results in a complete or partial re-shape of the tree then state at multiple tree members must be created/updated. In order to avoid packet loss this must be done synchronously. It will be discussed in future versions of this document how to achieve this.
5. Once the distribution tree is configured to replicate multicast traffic to the ETR the DTC Map-Replies (as specified in [I-D.farinacci-lisp-mr-signaling]) with the destination EID-prefix set to (parent-RLOC, ETR-RLOC).

When a LISP replication node signals its departure from the tree, the information stored at the LRND is updated accordingly. For ETRs, the state of the parent member must be updated as described in step 4. For RTRs both the state of the parent and its children must be updated however, such updates may result in packet loss. Moreover, certain optimization algorithms may result in a re-shape of the tree and as a consequence the state of multiple tree members must be created/updated according to the new topology. How to manage these updates with no packet loss will be discussed in future versions of this document.

## 5. Distribution Tree Optimization

Operators wishing to improve the performance of the distribution tree need to implement at least one topology discovery mechanism and choose a set of optimization algorithms. Due to the centralized group management, on-line switching between optimization algorithms



may be possible in accordance to the required performance. However, their use is dependent on the presence of overlay topological information. The following logical modules need to be implemented in order to support overlay optimizations with LISP-RE:

**Topology Discovery Coordinator:** It is in charge of organizing the topology measurements and building a database that stores the topological distances (i.e., a metric must be chosen) between overlay members.

**Distribution Tree Computation Unit:** It is a component that with the help of an algorithm or heuristic, given as input the topology of the overlay and a constraint, or constraint set, can compute an optimal, or close to optimal, degree-constrained minimum spanning tree that may be used for multicast content distribution.

Whether to implement the above modules in the ITR or in other network elements is the decision of the network administrator.

#### 5.1. Topology Discovery

The present document does not specify any topology discovery mechanisms. Both active and passive topology measurements could be used. A choice between the two, of the policy and admission control used or of the network element in charge of coordinating these measurements could be made in the future based on practical experience. Alternatively, precomputed network maps like the ones offered by [IPLANE] and/or out-of-band signalling may be used.

#### 5.2. Optimization Algorithm

The current document does not recommend an optimization algorithm. However, it provides as an example a low computation cost heuristic, which, in the scenarios simulated in [LCAST-TR], can produce latencies between the ITR and the multicast receivers close to unicast ones. Its choice is to be influenced by operational requirements and the hardware constraints of the equipment in charge of running it. Future experiments might result in a recommendation.

In what follows, we use the term "distance" when referring to a relative length or amplitude of a metric, observed on a path connecting two points, but when the exact nature of the metric is of no interest.

Considering as goal the delivery of content for delay sensitive applications, the function the algorithm minimizes is the maximum distance (e.g. latency or number of AS hops) from a multicast receiver to the ITR source. Notice that the reference is the

multicast receiver host and not an ETR. Thus, what matters in deciding a member's position in the distribution tree is not solely its distance to the ITR but also the number of multicast receivers it serves. Then, a router close to the source but serving few receivers might find itself lower in the distribution tree than another with a slightly higher distance to the source but with a larger receiver set. The algorithm optimizes the quality of experience for multicast receivers and not for tunnel routers.

The problem described above, that searches for a minimum average distance, degree-bounded spanning tree (MADDBST), can be formally stated as:

Definition: Given an undirected complete graph  $G=(V,E)$ , a designated vertex  $r$  belonging to  $V$ , for all vertices  $v$  in  $V$ , a degree bound  $d(v) \leq d_{max}$ ,  $d_{max}$  a positive integer, a vertex weight function  $c(v)$  with positive integer values, and an edge weight function  $w(e)$  with positive values, for all edges  $e$  in  $E$ . Let  $W(r,v,T)$  represent the cost of the path linking  $r$  and  $v$  in the spanning tree  $T$ . Find the spanning tree  $T$  of  $G$ , routed at  $r$ , satisfying that  $d(v) \leq d_{max}$  and the distance to the source per multicast receiver is minimized.

The heuristic used to solve this problem works by incrementally growing a tree, starting at the root node  $r$ , until it becomes a spanning tree. For each node  $v$ , not yet a tree member, it selects a potential parent node  $u$  in the tree  $T$ , such that the distance per receiver to  $r$ , is minimized. At each step, the node with the smallest metric value is added to the tree and the parent selection is redone. The pseudocode of the heuristic is provided in Appendix A.

[SHI] and [BAN] have previously defined and solved similar optimization problems. Shi et al. [SHI] also prove that a particular instance of the problem, where all vertices have weight 1, is NP-complete for degree constraints  $2 \leq d_{max} \leq |V|-1$ .

The algorithm can optimize an unicast overlay however, it should not be used to optimize multicast underlay delivery. As a result, if multicast is used as underlay between part of the overlay members, once one of the members of such Delivery Group is added to the distribution tree, the others should be marked as attached also. These nodes should receive multicast encapsulated multicast packets from the chosen node over the underlying multicast distribution tree.

Finally, since the RTRs do not replicate packets for multicast receiver hosts, prior to applying the MADDBST heuristic, a Minimum Spanning Tree (MST) algorithm should be used to compute the RTR

distribution tree. In this case, the MADDBST heuristic should start attaching ETRs having as input the tree resulting from MST.

## 6. Security Considerations

Security concerns for LISP-RE the same as for [I-D.ietf-lisp-multicast] and [I-D.farinacci-lisp-mr-signaling].

## 7. IANA Considerations

This memo includes no request to IANA.

## 8. Acknowledgements

TODO

## 9. References

### 9.1. Normative References

- [I-D.farinacci-lisp-lcaf]  
Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", draft-farinacci-lisp-lcaf-09 (work in progress), July 2012.
- [I-D.farinacci-lisp-mr-signaling]  
Farinacci, D., Spencer, T., and N. Napierala, "LISP Control-Plane Multicast Signaling", draft-farinacci-lisp-mr-signaling-00 (work in progress), July 2012.
- [I-D.farinacci-lisp-te]  
Farinacci, D., Lahiri, P., and M. Kowal, "LISP Traffic Engineering Use-Cases", draft-farinacci-lisp-te-01 (work in progress), July 2012.
- [I-D.ietf-lisp]  
Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-23 (work in progress), May 2012.
- [I-D.ietf-lisp-multicast]  
Farinacci, D., Meyer, D., Zwiebel, J., and S. Venaas, "LISP for Multicast Environments",

draft-ietf-lisp-multicast-14 (work in progress),  
February 2012.

- [RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas,  
"Protocol Independent Multicast - Sparse Mode (PIM-SM):  
Protocol Specification (Revised)", RFC 4601, August 2006.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for  
IP", RFC 4607, August 2006.

## 9.2. Informative References

- [BAN] Banerjee, S., Kommareddy, C., Kar, K., Bhattacharjee, B.,  
and S. Khuller, "Construction of an efficient overlay  
multicast infrastructure for real-time applications",  
INFOCOM , 2002.
- [IPLANE] Madhyastha, H., Katz-Bassett, E., Anderson, T.,  
Krishnamurthy, A., and A. Venkataramani, "iPlane: An  
Information Plane for Distributed Services", USENIX OSDI ,  
2009.
- [LCAST-TR] Coras, F., Cabellos, A., Domingo, J., Maino, F., and D.  
Farinacci, "Inter-Domain Multicast: LISP Edge Based  
Trees", Technical  
Report <http://personals.ac.upc.edu/fcoras/lcast-tr.pdf>,  
2012.
- [SHI] Shi, S., Turner, J., and M. Waldvogel, "Dimensioning  
server access bandwidth and multicast routing in overlay  
networks", NOSSDAV , 2001.

## Appendix A. MADDBST heuristic

```

INPUT: G = (V,E); r; dmax; w(u,v); c(v); u, v in V
OUTPUT: T

  FOREACH v in V DO
    delta(v) = w(r,v)/c(v);
    p(v) = r;
  END FOREACH

  T takes (U = {r}, D={});
  WHILE U != V DO
    LET u in U-V be the vertex with the smallest delta(u);
    U = U U {u}; L = L U {(p(u),u)};
    FOREACH v in V-U DO
      delta(v) = infinity;
      FOREACH u in U DO
        IF d(u) < dmax and
           W{r,u,T} + w(u,v)/c(v) < delta(v) THEN
          delta(v) = W{r,u,T} + w(u,v)/c(v);
          p(v) = u;
        END IF
      END FOR
    END FOR
  END WHILE

```

Figure 1

## Authors' Addresses

Florin Coras  
 Technical University of Catalonia  
 C/Jordi Girona, s/n  
 BARCELONA 08034  
 Spain

Email: fcoras@ac.upc.edu

Albert Cabellos-Aparicio  
 Technical University of Catalonia  
 C/Jordi Girona, s/n  
 BARCELONA 08034  
 Spain

Email: acabello@ac.upc.edu

Jordi Domingo-Pascual  
Technical University of Catalonia  
C/Jordi Girona, s/n  
BARCELONA 08034  
Spain

Email: [jordi.domingo@ac.upc.edu](mailto:jordi.domingo@ac.upc.edu)

Fabio Maino  
cisco Systems  
Tasman Drive  
San Jose, CA 95134  
USA

Email: [fmaino@cisco.com](mailto:fmaino@cisco.com)

Dino Farinacci  
cisco Systems  
Tasman Drive  
San Jose, CA 95134  
USA

Email: [dino@cisco.com](mailto:dino@cisco.com)



Network Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: January 10, 2013

D. Farinacci  
D. Meyer  
Cisco Systems  
J. Snijders  
InTouch N.V.  
July 9, 2012

LISP Canonical Address Format (LCAF)  
draft-farinacci-lisp-lcaf-10

Abstract

This draft defines a canonical address format encoding used in LISP control messages and in the encoding of lookup keys for the LISP Mapping Database System.

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 <http://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 January 10, 2013.

Copyright Notice

Copyright (c) 2012 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 (<http://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 . . . . .	4
3. LISP Canonical Address Format Encodings . . . . .	5
4. LISP Canonical Address Applications . . . . .	7
4.1. Segmentation using LISP . . . . .	7
4.2. Carrying AS Numbers in the Mapping Database . . . . .	8
4.3. Convey Application Specific Data . . . . .	9
4.4. Assigning Geo Coordinates to Locator Addresses . . . . .	10
4.5. Generic Database Mapping Lookups . . . . .	11
4.6. NAT Traversal Scenarios . . . . .	13
4.7. PETR Admission Control Functionality . . . . .	14
4.8. Multicast Group Membership Information . . . . .	16
4.9. Traffic Engineering using Re-encapsulating Tunnels . . . . .	17
4.10. Storing Security Data in the Mapping Database . . . . .	19
4.11. Source/Destination 2-Tuple Lookups . . . . .	20
4.12. Applications for AFI List Type . . . . .	21
4.12.1. Binding IPv4 and IPv6 Addresses . . . . .	21
4.12.2. Layer-2 VPNs . . . . .	22
4.12.3. ASCII Names in the Mapping Database . . . . .	22
4.12.4. Using Recursive LISP Canonical Address Encodings . . . . .	23
4.12.5. Compatibility Mode Use Case . . . . .	24
5. Security Considerations . . . . .	25
6. IANA Considerations . . . . .	26
7. References . . . . .	27
7.1. Normative References . . . . .	27
7.2. Informative References . . . . .	27
Appendix A. Acknowledgments . . . . .	28
Authors' Addresses . . . . .	29

## 1. Introduction

The LISP architecture and protocols [LISP] introduces two new numbering spaces, Endpoint Identifiers (EIDs) and Routing Locators (RLOCs) which are intended to replace most use of IP addresses on the Internet. To provide flexibility for current and future applications, these values can be encoded in LISP control messages using a general syntax that includes Address Family Identifier (AFI), length, and value fields.

Currently defined AFIs include IPv4 and IPv6 addresses, which are formatted according to code-points assigned in [AFI] as follows:

IPv4 Encoded Address:

```

      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
      +-----+-----+-----+-----+-----+-----+-----+-----+
      |                               AFI = 1                               |
      +-----+-----+-----+-----+-----+-----+-----+-----+
      | ... IPv4 Address |
      +-----+-----+-----+-----+-----+-----+-----+

```

IPv6 Encoded Address:

```

      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
      +-----+-----+-----+-----+-----+-----+-----+-----+
      |                               AFI = 2                               |
      +-----+-----+-----+-----+-----+-----+-----+-----+
      | ... IPv6 Address ... |
      +-----+-----+-----+-----+-----+-----+-----+-----+
      | ... IPv6 Address ... |
      +-----+-----+-----+-----+-----+-----+-----+-----+
      | ... IPv6 Address ... |
      +-----+-----+-----+-----+-----+-----+-----+-----+
      | ... IPv6 Address |
      +-----+-----+-----+-----+-----+-----+-----+

```

This document describes the currently-defined AFIs the LISP protocol uses along with their encodings and introduces the LISP Canonical Address Format (LCAF) that can be used to define the LISP-specific encodings for arbitrary AFI values.

## 2. Definition of Terms

**Address Family Identifier (AFI):** a term used to describe an address encoding in a packet. An address family currently defined for IPv4 or IPv6 addresses. See [AFI] and [RFC1700] for details. The reserved AFI value of 0 is used in this specification to indicate an unspecified encoded address where the length of the address is 0 bytes following the 16-bit AFI value of 0.

**Unspecified Address Format:**

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
|               AFI = 0               | <nothing follows AFI=0> |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

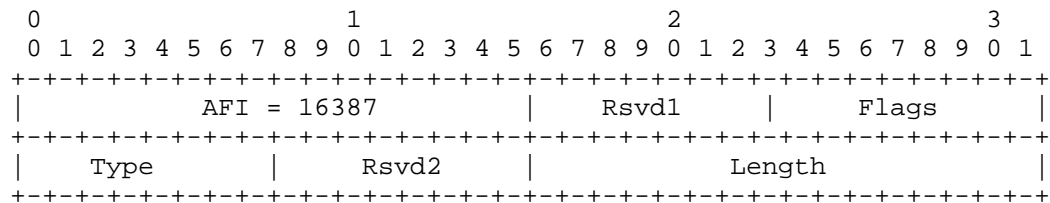
**Endpoint ID (EID):** a 32-bit (for IPv4) or 128-bit (for IPv6) value used in the source and destination address fields of the first (most inner) LISP header of a packet. The host obtains a destination EID the same way it obtains a destination address today, for example through a DNS lookup or SIP exchange. The source EID is obtained via existing mechanisms used to set a host's "local" IP address. 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.

**Routing Locator (RLOC):** the IPv4 or IPv6 address of an egress tunnel router (ETR). It is the output of a EID-to-RLOC mapping lookup. An EID maps to one or more RLOCs. Typically, RLOCs are numbered from topologically aggregatable blocks that are assigned to a site at each point to which it attaches to the global Internet; where the topology is defined by the connectivity of provider networks, RLOCs can be thought of as PA addresses. Multiple RLOCs can be assigned to the same ETR device or to multiple ETR devices at a site.

### 3. LISP Canonical Address Format Encodings

IANA has assigned AFI value 16387 (0x4003) to the LISP architecture and protocols. This specification defines the encoding format of the LISP Canonical Address (LCA).

The first 4 bytes of an LISP Canonical Address are followed by a variable length of fields:



Rsvd1: this 8-bit field is reserved for future use and MUST be transmitted as 0 and ignored on receipt.

Flags: this 8-bit field is for future definition and use. For now, set to zero on transmission and ignored on receipt.

Type: this 8-bit field is specific to the LISP Canonical Address formatted encodings, values are:

Type 0: Null Body Type

Type 1: AFI List Type

Type 2: Instance ID Type

Type 3: AS Number Type

Type 4: Application Data Type

Type 5: Geo Coordinates Type

Type 6: Opaque Key Type

Type 7: NAT-Traversal Type

Type 8: Nonce Locator Type

Type 9: Multicast Info Type

Type 10: Explicit Locator Path Type

Type 11: Security Key Type

Type 12: Source/Dest Key Type

Rsvd2: this 8-bit field is reserved for future use and MUST be transmitted as 0 and ignored on receipt.

Length: this 16-bit field is in units of bytes and covers all of the LISP Canonical Address payload, starting and including the byte after the Length field. So any LCAF encoded address will have a minimum length of 8 bytes when the Length field is 0. The 8 bytes include the AFI, Flags, Type, Reserved, and Length fields. When the AFI is not next to encoded address in a control message, then the encoded address will have a minimum length of 6 bytes when the Length field is 0. The 6 bytes include the Flags, Type, Reserved, and Length fields.

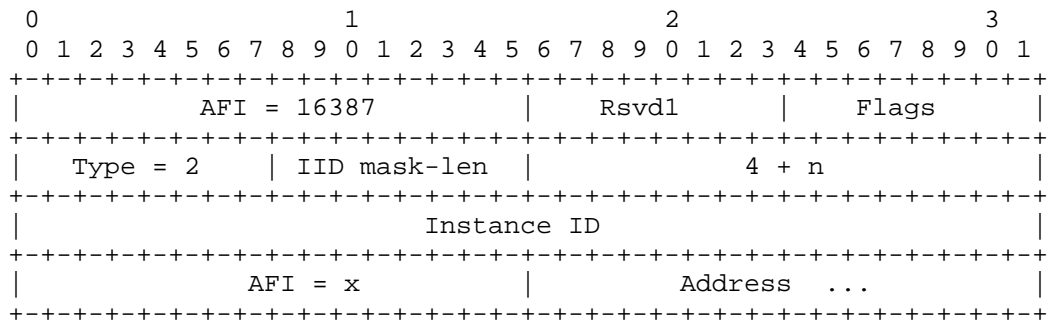
#### 4. LISP Canonical Address Applications

##### 4.1. Segmentation using LISP

When multiple organizations inside of a LISP site are using private addresses [RFC1918] as EID-prefixes, their address spaces must remain segregated due to possible address duplication. An Instance ID in the address encoding can aid in making the entire AFI based address unique.

Another use for the Instance ID LISP Canonical Address Format is when creating multiple segmented VPNs inside of a LISP site where keeping EID-prefix based subnets is desirable.

Instance ID LISP Canonical Address Format:



**IID mask-len:** if the AFI is set to 0, then this format is not encoding an extended EID-prefix but rather an instance-ID range where the 'IID mask-len' indicates the number of high-order bits used in the Instance ID field for the range.

**Length value n:** length in bytes of the AFI address that follows the Instance ID field including the AFI field itself.

**Instance ID:** the low-order 24-bits that can go into a LISP data header when the I-bit is set. See [LISP] for details.

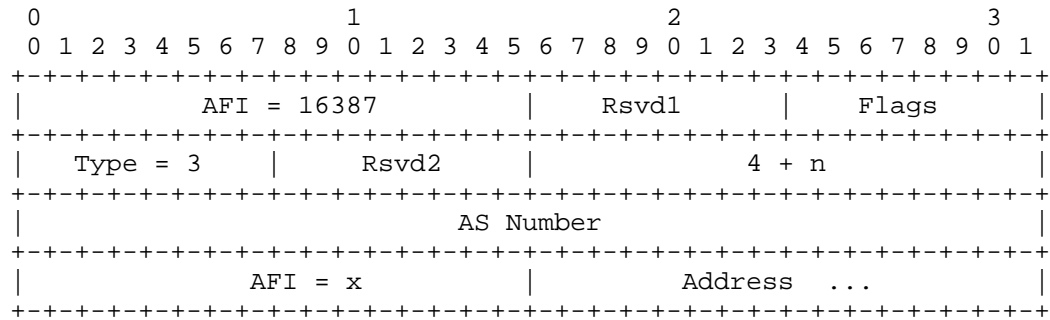
**AFI = x:** x can be any AFI value from [AFI].

This LISP Canonical Address Type can be used to encode either EID or RLOC addresses.

#### 4.2. Carrying AS Numbers in the Mapping Database

When an AS number is stored in the LISP Mapping Database System for either policy or documentation reasons, it can be encoded in a LISP Canonical Address.

AS Number LISP Canonical Address Format:



Length value n: length in bytes of the AFI address that follows the AS Number field including the AFI field itself.

AS Number: the 32-bit AS number of the autonomous system that has been assigned either the EID or RLOC that follows.

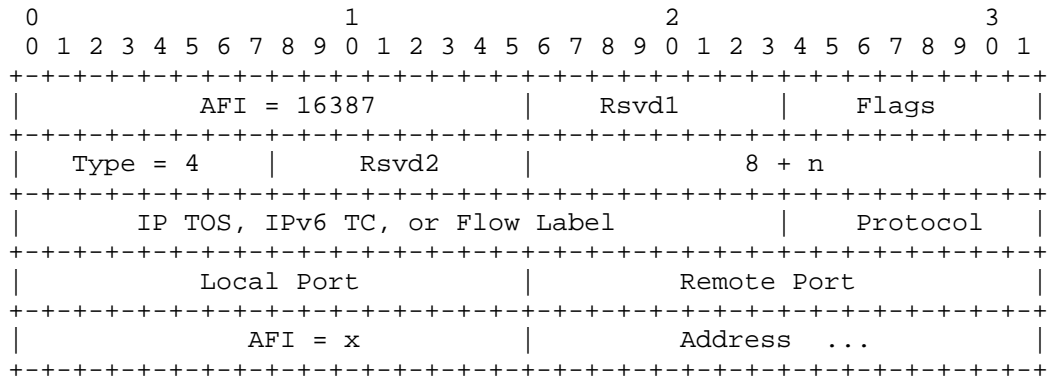
AFI = x: x can be any AFI value from [AFI].

The AS Number Canonical Address Type can be used to encode either EID or RLOC addresses. The former is used to describe the LISP-ALT AS number the EID-prefix for the site is being carried for. The latter is used to describe the AS that is carrying RLOC based prefixes in the underlying routing system.

### 4.3. Convey Application Specific Data

When a locator-set needs to be conveyed based on the type of application or the Per-Hop Behavior (PHB) of a packet, the Application Data Type can be used.

Application Data LISP Canonical Address Format:



Length value n: length in bytes of the AFI address that follows the 8-byte Application Data fields including the AFI field itself.

IP TOS, IPv6 TC, or Flow Label: this field stores the 8-bit IPv4 TOS field used in an IPv4 header, the 8-bit IPv6 Traffic Class or Flow Label used in an IPv6 header.

Local Port/Remote Port: these fields are from the TCP, UDP, or SCTP transport header.

AFI = x: x can be any AFI value from [AFI].

The Application Data Canonical Address Type is used for an EID encoding when an ITR wants a locator-set for a specific application. When used for an RLOC encoding, the ETR is supplying a locator-set for each specific application it has been configured to advertise.



#### 4.4. Assigning Geo Coordinates to Locator Addresses

If an ETR desires to send a Map-Reply describing the Geo Coordinates for each locator in its locator-set, it can use the Geo Coordinate Type to convey physical location information.

Geo Coordinate LISP Canonical Address Format:

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     AFI = 16387                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|   Type = 5   |   Rsvd2   |                                     12 + n   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|N|   Latitude Degrees   |   Minutes   |   Seconds   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|E|   Longitude Degrees   |   Minutes   |   Seconds   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Altitude                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     AFI = x   |   Address ...   |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Length value n: length in bytes of the AFI address that follows the 8-byte Longitude and Latitude fields including the AFI field itself.

N: When set to 1 means North, otherwise South.

Latitude Degrees: Valid values range from 0 to 90. degrees above or below the equator (northern or southern hemisphere, respectively).

Latitude Minutes: Valid values range from 0 to 59.

Latitude Seconds: Valid values range from 0 to 59.

E: When set to 1 means East, otherwise West.

Longitude Degrees: Value values are from 0 to 90 degrees right or left of the Prime Meridian.

Longitude Minutes: Valid values range from 0 to 59.

Longitude Seconds: Valid values range from 0 to 59.

Altitude: Height relative to sea level in meters. This is a signed integer meaning that the altitude could be below sea level. A value of 0x7fffffff indicates no Altitude value is encoded.

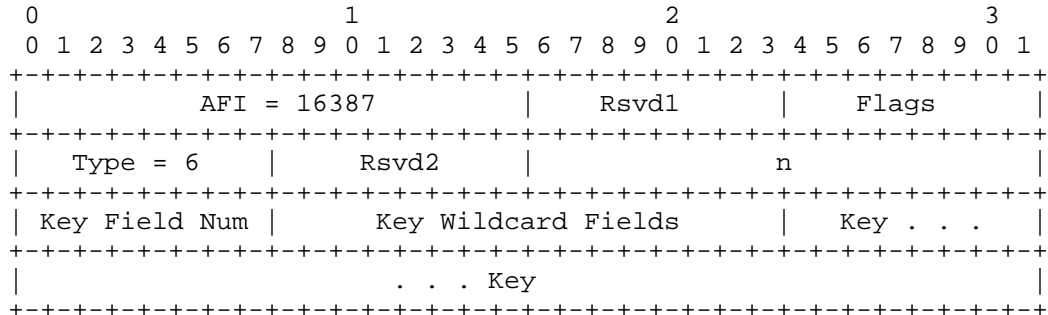
AFI = x: x can be any AFI value from [AFI].

The Geo Coordinates Canonical Address Type can be used to encode either EID or RLOC addresses. When used for EID encodings, you can determine the physical location of an EID along with the topological location by observing the locator-set.

#### 4.5. Generic Database Mapping Lookups

When the LISP Mapping Database system holds information accessed by a generic formatted key (where the key is not the usual IPv4 or IPv6 address), an opaque key may be desirable.

Opaque Key LISP Canonical Address Format:



Length value n: length in bytes of the type's payload. The value n is the number of bytes that follow this Length field.

Key Field Num: the number of fields (minus 1) the key can be broken up into. The width of the fields are fixed length. So for a key size of 8 bytes, with a Key Field Num of 4 allows 4 fields of 2 bytes in length. Valid values for this field range from 0 to 15 supporting a maximum of 16 field separations.

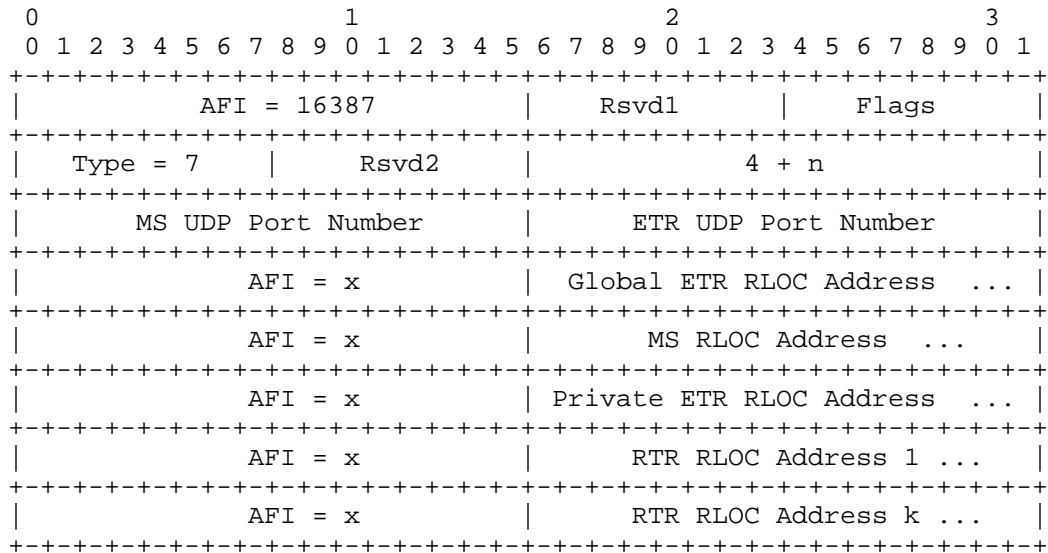
Key Wildcard Fields: describes which fields in the key are not used as part of the key lookup. This wildcard encoding is a bitfield. Each bit is a don't-care bit for a corresponding field in the key. Bit 0 (the low-order bit) in this bitfield corresponds the first field, right-justified in the key, bit 1 the second field, and so on. When a bit is set in the bitfield it is a don't-care bit and should not be considered as part of the database lookup. When the entire 16-bits is set to 0, then all bits of the key are used for the database lookup.

Key: the variable length key used to do a LISP Database Mapping lookup. The length of the key is the value n (shown above) minus 3.

## 4.6. NAT Traversal Scenarios

When a LISP system is conveying global address and mapped port information when traversing through a NAT device, the NAT-Traversal LCAF Type is used. See [LISP-NATT] for details.

NAT-Traversal Canonical Address Format:



Length value n: length in bytes of the AFI addresses that follows the UDP Port Number field including the AFI fields themselves.

MS UDP Port Number: this is the UDP port number of the Map-Server and is set to 4342.

ETR UDP Port Number: this is the port number returned to a LISP system which was copied from the source port from a packet that has flowed through a NAT device.

AFI = x: x can be any AFI value from [AFI].

Global ETR RLOC Address: this is an address known to be globally unique built by NAT-traversal functionality in a LISP router.

MS RLOC Address: this is the address of the Map-Server used in the destination RLOC of a packet that has flowed through a NAT device.

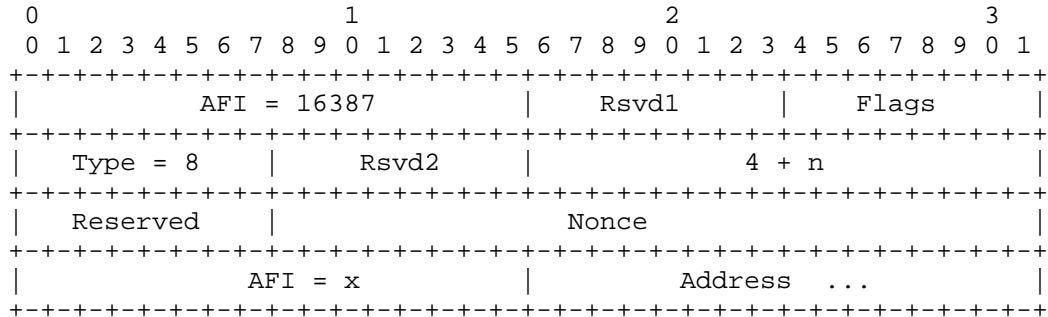
Private ETR RLOC Address: this is an address known to be a private address inserted in this LCAF format by a LISP router that resides on the private side of a NAT device.

RTR RLOC Address: this is an encapsulation address used by an ITR or PITR which resides behind a NAT device. This address is known to have state in a NAT device so packets can flow from it to the LISP ETR behind the NAT. There can be one or more NTR addresses supplied in these set of fields. The number of NTRs encoded is determined by the LCAF length field. When there are no NTRs supplied, the NTR fields can be omitted and reflected by the LCAF length field or an AFI of 0 can be used to indicate zero NTRs encoded.

#### 4.7. PETR Admission Control Functionality

When a public PETR device wants to verify who is encapsulating to it, it can check for a specific nonce value in the LISP encapsulated packet. To convey the nonce to admitted ITRs or PITRs, this LCAF format is used in a Map-Register or Map-Reply locator-record.

## Nonce Locator Canonical Address Format:



Length value n: length in bytes of the AFI address that follows the Nonce field including the AFI field itself.

Reserved: must be set to zero and ignore on receipt.

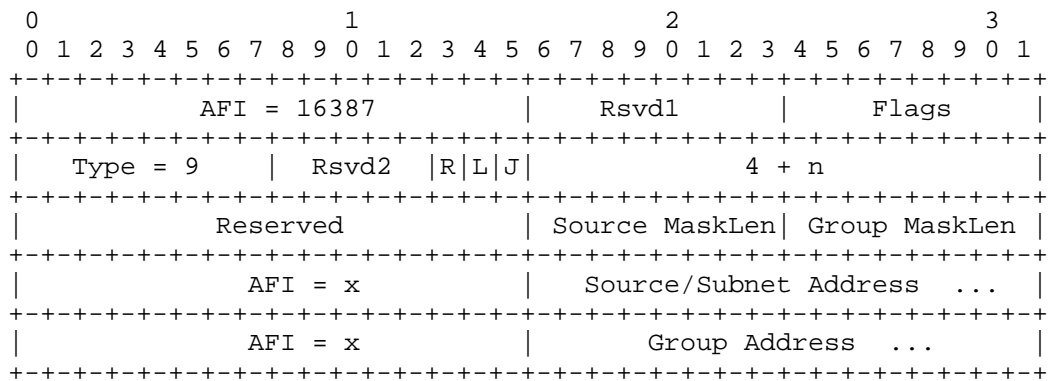
Nonce: this is a nonce value returned by an ETR in a Map-Reply locator-record to be used by an ITR or PITR when encapsulating to the locator address encoded in the AFI field of this LCAF type.

AFI = x: x can be any AFI value from [AFI].

## 4.8. Multicast Group Membership Information

Multicast group information can be published in the mapping database so a lookup on an EID based group address can return a replication list of group addresses or a unicast addresses for single replication or multiple head-end replications. This LCAF encoding can be used to send broadcast packets to all members of a subnet when each EIDs are away from their home subnet location.

Multicast Info Canonical Address Format:



Length value n: length in bytes of fields that follow.

Reserved: must be set to zero and ignore on receipt.

R-bit: this is the RP-bit that represents PIM (S,G,RP-bit) multicast state. This bit can be set for Joins (when the J-bit is set) or for Leaves (when the L-bit is set). See [LISP-MRSIG] for more usage details.

L-bit: this is the Leave-Request bit and is used when this LCAF type is present in the destination EID-prefix field of a Map-Request. See [LISP-MRSIG] for details.

J-bit: this is the Join-Request bit and is used when this LCAF type is present in the destination EID-prefix field of a Map-Request. See [LISP-MRSIG] for details. The J-bit MUST not be set when the L-bit is also set in the same LCAF block. A receiver should not take any specific Join or Leave action when both bits are set.

Source MaskLen: the mask length of the source prefix that follows.

Group MaskLen:    the mask length of the group prefix that follows.

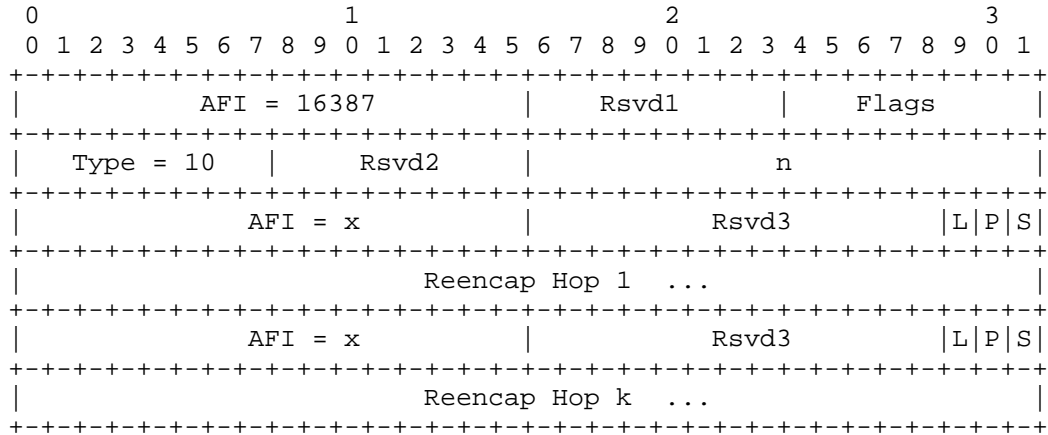
AFI = x:    x can be any AFI value from [AFI]. When a specific AFI has its own encoding of a multicast address, this field must be either a group address or a broadcast address.

#### 4.9. Traffic Engineering using Re-encapsulating Tunnels

For a given EID lookup into the mapping database, this LCAF format can be returned to provide a list of locators in an explicit re-encapsulation path. See [LISP-TE] for details.



## Explicit Locator Path (ELP) Canonical Address Format:



Length value n: length in bytes of fields that follow.

AFI = x: x can be any AFI value from [AFI]. When a specific AFI has its own encoding of a multicast address, this field must be either a group address or a broadcast address.

Lookup bit (L): this is the Lookup bit used to indicate to the user of the ELP to not use this address for encapsulation but to look it up in the mapping database system to obtain an encapsulating RLOC address.

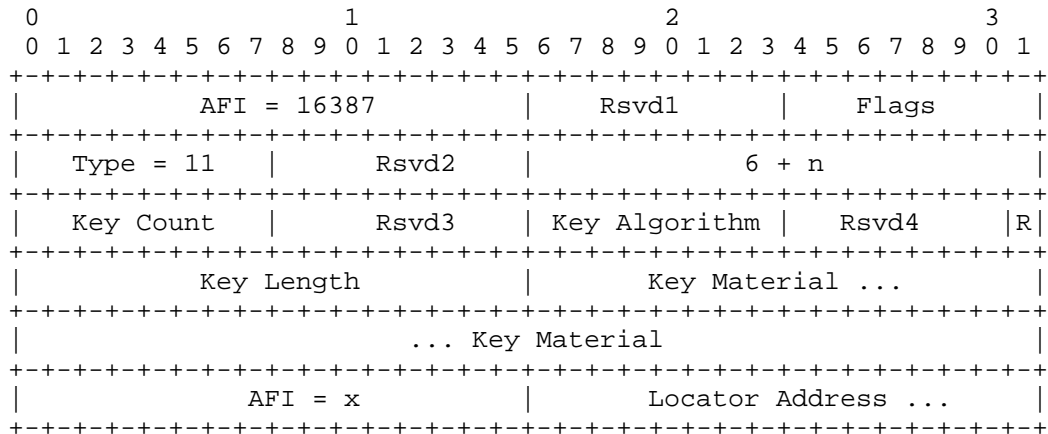
RLOC-Probe bit (P): this is the RLOC-probe bit which means the Reencap Hop allows RLOC-probe messages to be sent to it. When the R-bit is set to 0, RLOC-probes must not be sent. When a Reencap Hop is an anycast address then multiple physical Reencap Hops are using the same RLOC address. In this case, RLOC-probes are not needed because when the closest RLOC address is not reachable another RLOC address can be reachable.

Strict bit (S): this is the strict bit which means the associated Reencap Hop is required to be used. If this bit is 0, the reencapsulator can skip this Reencap Hop and go to the next one in the list.

## 4.10. Storing Security Data in the Mapping Database

When a locator in a locator-set has a security key associated with it, this LCAF format will be used to encode key material. See [LISP-DDT] for details.

Security Key Canonical Address Format:



Length value n: length in bytes of fields that start with the Key Material field.

Key Count: the Key Count field declares the number of Key sections included in this LCAF.

Key Algorithm: the Algorithm field identifies the key's cryptographic algorithm and specifies the format of the Public Key field.

R bit: this is the revoke bit and, if set, it specifies that this Key is being Revoked.

Key Length: this field determines the length in bytes of the Key Material field.

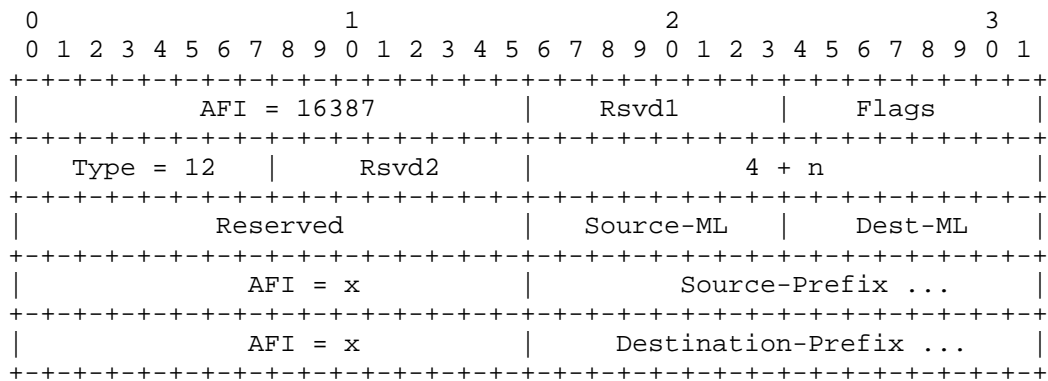
Key Material: the Key Material field stores the key material. The format of the key material stored depends on the Key Algorithm field.

AFI = x: x can be any AFI value from [AFI]. This is the locator address that owns the encoded security key.

## 4.11. Source/Destination 2-Tuple Lookups

When both a source and destination address of a flow needs consideration for different locator-sets, this 2-tuple key is used in EID fields in LISP control messages. When the Source/Dest key is registered to the mapping database, it can be encoded as a source-prefix and destination-prefix. When the Source/Dest is used as a key for a mapping database lookup the source and destination come from a data packet.

Source/Dest Key Canonical Address Format:



Length value n: length in bytes of fields that follow.

Reserved: must be set to zero and ignore on receipt.

Source-ML: the mask length of the source prefix that follows.

Dest-ML: the mask length of the destination prefix that follows.

AFI = x: x can be any AFI value from [AFI]. When a specific AFI has its own encoding of a multicast address, this field must be either a group address or a broadcast address.

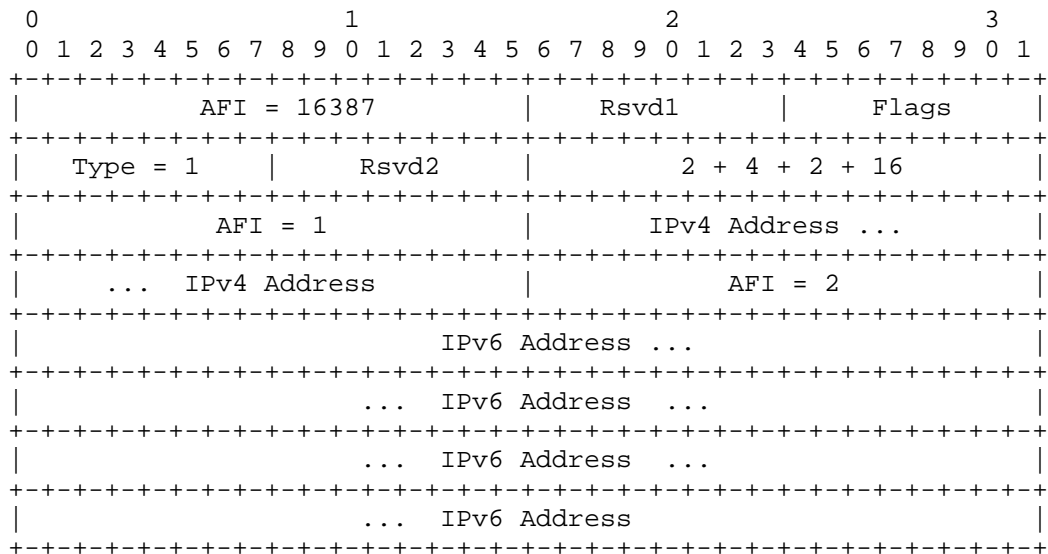
Refer to [LISP-TE] for usage details.

## 4.12. Applications for AFI List Type

## 4.12.1. Binding IPv4 and IPv6 Addresses

When header translation between IPv4 and IPv6 is desirable a LISP Canonical Address can use the AFI List Type to carry multiple AFIs in one LCA AFI.

Bounded Address LISP Canonical Address Format:



Length: length in bytes is fixed at 24 when IPv4 and IPv6 AFI encoded addresses are used.

This type of address format can be included in a Map-Request when the address is being used as an EID, but the Mapping Database System lookup destination can use only the IPv4 address. This is so a Mapping Database Service Transport System, such as LISP-ALT [ALT], can use the Map-Request destination address to route the control message to the desired LISP site.

## 4.12.2. Layer-2 VPNs

When MAC addresses are stored in the LISP Mapping Database System, the AFI List Type can be used to carry AFI 6.

MAC Address LISP Canonical Address Format:

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     AFI = 16387                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      Type = 1      |      Rsvd2      |                                     2 + 6      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     AFI = 6                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     ... Layer-2 MAC Address ...      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     ... Layer-2 MAC Address      |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Length: length in bytes is fixed at 8 when MAC address AFI encoded addresses are used.

This address format can be used to connect layer-2 domains together using LISP over an IPv4 or IPv6 core network to create a layer-2 VPN. In this use-case, a MAC address is being used as an EID, and the locator-set that this EID maps to can be an IPv4 or IPv6 RLOCs, or even another MAC address being used as an RLOC.

## 4.12.3. ASCII Names in the Mapping Database

If DNS names or URIs are stored in the LISP Mapping Database System, the AFI List Type can be used to carry an ASCII string where it is delimited by length 'n' of the LCAF Length encoding.

ASCII LISP Canonical Address Format:

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     AFI = 16387                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      Type = 1      |      Rsvd2      |                                     2 + n      |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     AFI = 17                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     DNS Name or URI ...      |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Length value n: length in bytes AFI=17 field and the null-terminated ASCII string (the last byte of 0 is included).

#### 4.12.4. Using Recursive LISP Canonical Address Encodings

When any combination of above is desirable, the AFI List Type value can be used to carry within the LCA AFI another LCA AFI.

Recursive LISP Canonical Address Format:

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								
AFI = 16387										Rsvd1										Flags																			
Type = 1										Rsvd2										4 + 8 + 2 + 4																			
AFI = 16387										Rsvd1										Flags																			
Type = 4										Rsvd2										12																			
IP TOS, IPv6 QoS or Flow Label															Protocol																								
Local Port															Remote Port																								
AFI = 1										IPv4 Address ...																													
... IPv4 Address																																							

Length: length in bytes is fixed at 18 when an AFI=1 IPv4 address is included.

This format could be used by a Mapping Database Transport System, such as LISP-ALT [ALT], where the AFI=1 IPv4 address is used as an EID and placed in the Map-Request destination address by the sending LISP system. The ALT system can deliver the Map-Request to the LISP destination site independent of the Application Data Type AFI payload values. When this AFI is processed by the destination LISP site, it can return different locator-sets based on the type of application or level of service that is being requested.

## 4.12.5. Compatibility Mode Use Case

A LISP system should use the AFI List Type format when sending to LISP systems that do not support a particular LCAF Type used to encode locators. This allows the receiving system to be able to parse a locator address for encapsulation purposes. The list of AFIs in an AFI List LCAF Type has no semantic ordering and a receiver should parse each AFI element no matter what the ordering.

Compatibility Mode Address Format:

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								
AFI = 16387										Rsvd1										Flags																			
Type = 1										Rsvd2										22 + 6																			
AFI = 16387										Rsvd1										Flags																			
Type = 5										Rsvd2										12 + 2																			
N	Latitude Degrees										Minutes										Seconds																		
E	Longitude Degrees										Minutes										Seconds																		
Altitude																																							
AFI = 0										AFI = 1																													
IPv4 Address																																							

If a system does not recognize the Geo Coordinate LCAF Type that is accompanying a locator address, an encoder can include the Geo Coordinate LCAF Type embedded in an AFI List LCAF Type where the AFI in the Geo Coordinate LCAF is set to 0 and the AFI encoded next in the list is encoded with a valid AFI value to identify the locator address.

A LISP system is required to support the AFI List LCAF Type to use this procedure. It would skip over 10 bytes of the Geo Coordinate LCAF Type to get to the locator address encoding (an IPv4 locator address). A LISP system that does support the Geo Coordinate LCAF Type can support parsing the locator address within the Geo Coordinate LCAF encoding or in the locator encoding that follows in the AFI List LCAF.

## 5. Security Considerations

There are no security considerations for this specification. The security considerations are documented for the protocols that use LISP Canonical Addressing. Refer to the those relevant specifications.



## 6. IANA Considerations

The Address Family AFI definitions from [AFI] only allocate code-points for the AFI value itself. The length of the address or entity that follows is not defined and is implied based on conventional experience. Where the LISP protocol uses LISP Canonical Addresses specifically, the address length definitions will be in this specification and take precedent over any other specification.

An IANA Registry for LCAF Type values will be created. The values that are considered for use by the main LISP specification [LISP] will be in the IANA Registry. Other Type values used for experimentation will be defined and described in this document.

## 7. References

### 7.1. Normative References

- [RFC1700] Reynolds, J. and J. Postel, "Assigned Numbers", RFC 1700, October 1994.
- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

### 7.2. Informative References

- [AFI] IANA, "Address Family Identifier (AFIs)", ADDRESS FAMILY NUMBERS <http://www.iana.org/numbers.html>, Febuary 2007.
- [ALT] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "LISP Alternative Topology (LISP+ALT)", draft-ietf-lisp-alt-06.txt (work in progress), March 2011.
- [LISP] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-12.txt (work in progress), April 2011.
- [LISP-DDT] Fuller, V., Lewis, D., and V. Ermagan, "LISP Delegated Database Tree", draft-fuller-lisp-ddt-01.txt (work in progress).
- [LISP-MRSIG] Farinacci, D. and M. Napierala, "LISP Control-Plane Multicast Signaling", draft-farinacci-lisp-mr-signaling-00.txt (work in progress).
- [LISP-NATT] Ermagan, V., Farinacci, D., Lewis, D., Skriver, J., Maino, F., and C. White, "NAT traversal for LISP", draft-ermagan-lisp-nat-traversal-00.txt (work in progress).
- [LISP-TE] Farinacci, D., Lahiri, P., and M. Kowal, "LISP Traffic Engineering Use-Cases", draft-farinacci-lisp-te-01.txt (work in progress).

## Appendix A. Acknowledgments

The authors would like to thank Vince Fuller, Gregg Schudel, Jesper Skriver, Luigi Iannone, and Isidor Kouvelas for their technical and editorial commentary.

The authors would like to thank Victor Moreno for discussions that lead to the definition of the Multicast Info LCAF type.

The authors would like to thank Parantap Lahiri and Michael Kowal for discussions that lead to the definition of the Explicit Locator Path (ELP) LCAF type.

The authors would like to thank Fabio Maino and Vina Ermagan for discussions that lead to the definition of the Security Key LCAF type.

Thanks also goes to Terry Manderson for assistance obtaining a LISP AFI value from IANA.

Authors' Addresses

Dino Farinacci  
cisco Systems  
Tasman Drive  
San Jose, CA 95134  
USA

Email: dino@cisco.com

Dave Meyer  
cisco Systems  
170 Tasman Drive  
San Jose, CA  
USA

Email: dmm@cisco.com

Job Snijders  
InTouch N.V.  
Middenweg 76  
1097 BS Amsterdam  
The Netherlands

Email: job@instituut.net



Internet Engineering Task Force  
Internet-Draft  
Intended status: Experimental  
Expires: January 4, 2013

D. Farinacci  
cisco Systems  
P. Lahiri  
Microsoft Corporation  
M. Kowal  
cisco Systems  
July 3, 2012

LISP Traffic Engineering Use-Cases  
draft-farinacci-lisp-te-01

Abstract

This document describes how LISP reencapsulating tunnels can be used for Traffic Engineering purposes. The mechanisms described in this document require no LISP protocol changes but do introduce a new locator (RLOC) encoding. The Traffic Engineering features provided by these LISP mechanisms can span intra-domain, inter-domain, or combination of both.

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 <http://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 January 4, 2013.

Copyright Notice

Copyright (c) 2012 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 (<http://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. Requirements Language . . . . .	3
2. Introduction . . . . .	4
3. Definition of Terms . . . . .	5
4. Overview . . . . .	7
5. Explicit Locator Paths . . . . .	9
5.1. ELP Re-optimization . . . . .	10
5.2. Using Recursion . . . . .	10
5.3. ELP Selection based on Class of Service . . . . .	11
5.4. Packet Loop Avoidance . . . . .	12
6. Service Chaining . . . . .	13
7. RLOC Probing by RTRs . . . . .	14
8. Interworking Considerations . . . . .	15
9. Multicast Considerations . . . . .	16
10. Security Considerations . . . . .	18
11. IANA Considerations . . . . .	19
12. References . . . . .	20
12.1. Normative References . . . . .	20
12.2. Informative References . . . . .	20
Appendix A. Acknowledgments . . . . .	22
Appendix B. Document Change Log . . . . .	23
B.1. Changes to draft-farinacci-lisp-te-01.txt . . . . .	23
B.2. Changes to draft-farinacci-lisp-te-00.txt . . . . .	23
Authors' Addresses . . . . .	24

## 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 RFC 2119 [RFC2119].



## 2. Introduction

This document describes the Locator/Identifier Separation Protocol (LISP), which provides a set of functions for routers to exchange information used to map from non globally routeable Endpoint Identifiers (EIDs) to routeable Routing Locators (RLOCs). It also defines a mechanism for these LISP routers to encapsulate IP packets addressed with EIDs for transmission across the Internet that uses RLOCs for routing and forwarding.

When LISP routers encapsulate packets to other LISP routers, the path stretch is typically 1, meaning the packet travels on a direct path from the encapsulating ITR to the decapsulating ETR at the destination site. The direct path is determined by the underlying routing protocol and metrics it uses to find the shortest path.

This specification will examine how reencapsulating tunnels [LISP] can be used so a packet can take an administratively specified path, a congestion avoidance path, a failure recovery path, or multiple load-shared paths, as it travels from ITR to ETR. By introducing an Explicit Locator Path (ELP) locator encoding [LISP-LCAF], an ITR can encapsulate a packet to a Reencapsulating Tunnel Router (RTR) which decapsulates the packet, then encapsulates it to the next locator in the ELP.

### 3. Definition of Terms

**Endpoint ID (EID):** An EID is a 32-bit (for IPv4) or 128-bit (for IPv6) value used 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 an destination address today, for example through a Domain Name System (DNS) [RFC1034] lookup or Session Invitation 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 globally 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. EIDs MUST NOT be used as LISP RLOCs. 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 global routing system. In theory, the bit string that represents an EID for one device can represent an RLOC for a different device. As the architecture is realized, if a given bit string is both an RLOC and an EID, it must refer to the same entity in both cases. When used in discussions with other Locator/ID separation proposals, a LISP EID will be called a "LEID". Throughout this document, any references to "EID" refers to an LEID.

**Routing Locator (RLOC):** A RLOC is an IPv4 [RFC0791] or IPv6 [RFC2460] address of an egress tunnel router (ETR). A RLOC is the output of an EID-to-RLOC mapping lookup. An EID maps to one or more RLOCs. Typically, RLOCs are numbered from topologically-aggregatable blocks that are assigned to a site at each point to which it attaches to the global Internet; where the topology is defined by the connectivity of provider networks, RLOCs can be thought of as PA addresses. Multiple RLOCs can be assigned to the same ETR device or to multiple ETR devices at a site.

**Reencapsulating Tunnel Router (RTR):** An RTR is a router that acts as an ETR (or PETR) by decapsulating packets where the destination address in the "outer" IP header is one of its own RLOCs. Then acts as an ITR (or PITR) by making a decision where to encapsulate the packet based on the next locator in the ELP towards the final destination ETR.

**Explicit Locator Path (ELP):** The ELP is an explicit list of RLOCs for each RTR a packet must travel to along its path toward a final destination ETR (or PETR). The list is a strict ordering where each RLOC in the list is visited. However, the path from one RTR to another is determined by the underlying routing protocol and how the infrastructure assigns metrics and policies for the path.

**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. Any references to tunnels in this specification refers to dynamic encapsulating tunnels and they are never statically configured.

**Reencapsulating Tunnels:** Reencapsulating tunneling occurs when an ETR removes a LISP header, then acts as an ITR to prepend another LISP header. Doing this allows a packet to be re-routed by the reencapsulating router without adding the overhead of additional tunnel headers. Any references to tunnels in this specification refers to dynamic encapsulating tunnels and they are never statically configured. When using multiple mapping database systems, care must be taken to not create reencapsulation loops through misconfiguration.

#### 4. Overview

Typically, a packet's path from source EID to destination EID travels through the locator core via the encapsulating ITR directly to the decapsulating ETR as the following diagram illustrates:

Legend:

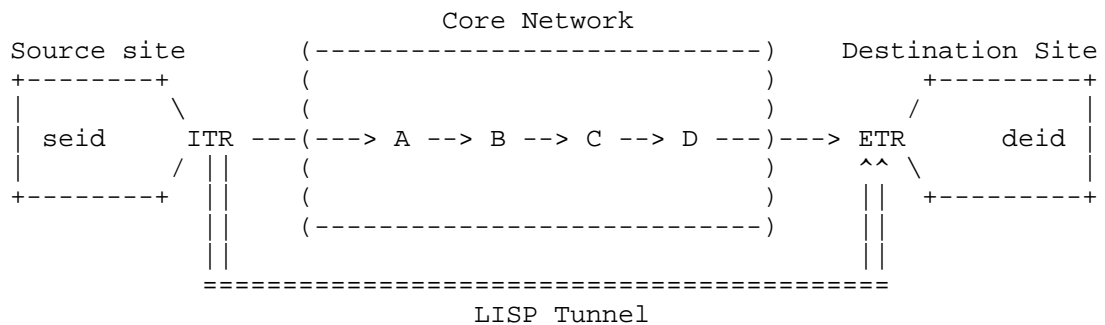
seid: Packet is originated by source EID 'seid'.

deid: Packet is consumed by destination EID 'deid'.

A,B,C,D : Core routers in different ASes.

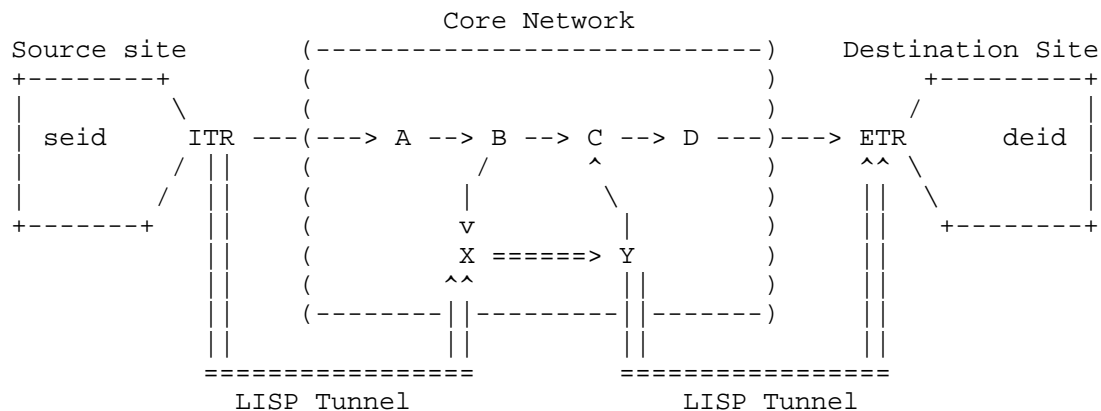
---> : The physical topological path between two routers.

==> : A multi-hop LISP dynamic tunnel between LISP routers.



#### Typical Data Path from ITR to ETR

Let's introduce RTRs 'X' and 'Y' so that, for example, if it is desirable to route around the path from B to C, one could provide an ELP of (X,Y,etr):



ELP tunnel path ITR ==> X, then X ==> Y, and then Y ==> ETR

There are various reasons why the path from 'seid' to 'deid' may want to avoid the path from B to C. To list a few:

- o There may not be sufficient capacity provided by the networks that connect B and C together.
- o There may be a policy reason to avoid the ASes that make up the path between B and C.
- o There may be a failure on the path between B and C which makes the path unreliable.
- o There may be monitoring or traffic inspection resources close to RTRs X and Y that do network accounting or measurement.
- o There may be a chain of services performed at RTRs X and Y regardless if the path from ITR to ETR is through B and C.

## 5. Explicit Locator Paths

The notation for a general formatted ELP is (x, y, etr) which represents the list of RTRs a packet SHOULD travel through to reach the final tunnel hop to the ETR.

The procedure for using an ELP at each tunnel hop is as follows:

1. The ITR will retrieve the ELP from the mapping database.
2. The ITR will encapsulate the packet to RLOC 'x'.
3. The RTR with RLOC 'x' will decapsulate the packet. It will use the decapsulated packet's destination address as a lookup into the mapping database to retrieve the ELP.
4. RTR 'x' will encapsulate the packet to RTR with RLOC 'y'.
5. The RTR with RLOC 'y' will decapsulate the packet. It will use the decapsulated packet's destination address as a lookup into the mapping database to retrieve the ELP.
6. RTR 'y' will encapsulate the packet on the final tunnel hop to ETR with RLOC 'etr'.
7. The ETR will decapsulate the packet and deliver the packet to the EID inside of its site.

The specific format for the ELP can be found in [LISP-LCAF]. It is defined that an ELP will appear as a single encoded locator in a locator-set. Say for instance, we have a mapping entry for EID-prefix 10.0.0.0/8 that is reachable via 4 locators. Two locators are being used as active/active and the other two are used as active/active if the first two go unreachable (as noted by the priority assignments below). This is what the mapping entry would look like:

```
EID-prefix:    10.0.0.0/8
Locator-set:   ETR-A: priority 1, weight 50
               ETR-B: priority 1, weight 50
               ETR-C: priority 2, weight 50
               ETR-D: priority 2, weight 50
```

If an ELP is going to be used to have a policy path to ETR-A and possibly another policy path to ETR-B, the locator-set would be encoded as follows:

```
EID-prefix: 10.0.0.0/8
Locator-set: (x, y, ETR-A): priority 1, weight 50
              (q, r, ETR-B): priority 1, weight 50
              ETR-C:         priority 2, weight 50
              ETR-D:         priority 2, weight 50
```

The mapping entry with ELP locators is registered to the mapping database system just like any other mapping entry would. The registration is typically performed by the ETR(s) that are assigned and own the EID-prefix. That is, the destination site makes the choice of the RTRs in the ELP. However, it may be common practice for a provisioning system to program the mapping database with ELPs.

Another case where a locator-set can be used for flow-based load-sharing across multiple paths to the same destination site:

```
EID-prefix: 10.0.0.0/8
Locator-set: (x, y, ETR-A): priority 1, weight 75
              (q, r, ETR-A): priority 1, weight 25
```

Using this mapping entry, an ITR would load split 75% of the EID flows on the (x, y, ETR-A) ELP path and 25% of the EID flows on the (q, r, ETR-A) ELP path. If any of the ELPs go down, then the other can take 100% of the load.

#### 5.1. ELP Re-optimization

ELP re-optimization is a process of changing the RLOCs of an ELP due to underlying network change conditions. Just like when there is any locator change for a locator-set, the procedures from the main LISP specification [LISP] are followed.

When a RLOC from an ELP is changed, Map-Notify messages [LISP-MS] can be used to inform the existing RTRs in the ELP so they can do a lookup to obtain the latest version of the ELP. Map-Notify messages can also be sent to new RTRs in an ELP so they can get the ELP in advance to receiving packets that will use the ELP. This can minimize packet loss during mapping database lookups in RTRs.

#### 5.2. Using Recursion

In the previous examples, we showed how an ITR encapsulates using an ELP of (x, y, etr). When a packet is encapsulated from the ITR to RTR 'x', the RTR may want a policy path to RTR 'y' and run another level of reencapsulating tunnels for packets destined to RTR 'y'. In this case, RTR 'x' does not decapsulate packets from the ITR, but rather performs a mapping database lookup on the address 'y'. This

can be done when using a public or private mapping database. The decision to use address 'y' as an encapsulation address versus a lookup address is based on the L-bit setting for 'y' in the ELP entry. The decision and policy of ELP encodings are local to the entity which registers the EID-prefix associated with the ELP.

Another example of recursion is when the ITR uses the ELP (x, y, etr) to first prepend a header with a destination RLOC of the ETR and then prepend another header and encapsulate the packet to RTR 'x'. When RTR 'x' decapsulates the packet, rather than doing a mapping database lookup on RTR 'y' the last example showed, instead RTR 'x' does a mapping database lookup on ETR 'etr'. In this scenario, RTR 'x' can choose an ELP from the locator-set by considering the source RLOC address of the ITR versus considering the source EID.

This additional level of recursion also brings advantages for the provider of RTR 'x' to store less state. Since RTR 'x' does not need to look at the inner most header, it does not need to store EID state. It only stores an entry for RTR 'y' which many EID flows could share for scaling benefits. The locator-set for entry 'y' could either be a list of typical locators, a list of ELPs, or combination of both. Another advantage is that packet load-splitting can be accomplished by examining the source of a packet. If the source is an ITR versus the source being the last-hop of an ELP the last-hop selected, different forwarding paths can be used.

### 5.3. ELP Selection based on Class of Service

Paths to an ETR may want to be selected based on different classes of service. Packets from a set of sources that have premium service can use ELP paths that are less congested where normal sources use ELP paths that compete for less resources or use longer paths for best effort service.

Using source/destination lookups into the mapping database can yield different ELPs. So for example, a premium service flow with (source=1.1.1.1, dest=10.1.1.1) can be described by using the following mapping entry:

```
EID-prefix:    (1.0.0.0/8, 10.0.0.0/8)
Locator-set:   (x, y, ETR-A): priority 1, weight 50
               (q, r, ETR-A): priority 1, weight 50
```

And all other best-effort sources would use different mapping entry described by:



```
EID-prefix:    (0.0.0.0/0, 10.0.0.0/8)
Locator-set:   (x, x', y, y', ETR-A): priority 1, weight 50
               (q, q', r, r', ETR-A): priority 1, weight 50
```

If the source/destination lookup is coupled with recursive lookups, then an ITR can encapsulate to the ETR, prepending a header that selects source address ITR-1 based on the premium class of service source, or selects source address ITR-2 for best-effort sources with normal class of service. The ITR then does another lookup in the mapping database on the prepended header using lookup key (source=ITR-1, dest=10.1.1.1) that returns the following mapping entry:

```
EID-prefix:    (ITR-1, 10.0.0.0/8)
Locator-set:   (x, y, ETR-A): priority 1, weight 50
               (q, r, ETR-A): priority 1, weight 50
```

And all other sources would use different mapping entry with a lookup key of (source=ITR-2, dest=10.1.1.1):

```
EID-prefix:    (ITR-2, 10.0.0.0/8)
Locator-set:   (x, x', y, y', ETR-A): priority 1, weight 50
               (q, q', r, r', ETR-A): priority 1, weight 50
```

This will scale the mapping system better by having fewer source/destination combinations. Refer to the Source/Dest LCAF type described in [LISP-LCAF] for encoding EIDs in Map-Request and Map-Register messages.

#### 5.4. Packet Loop Avoidance

An ELP that is first used by an ITR must be inspected for encoding loops. If any RLOC appears twice in the ELP, it MUST not be used.

Since it is expected that multiple mapping systems will be used, there can be a loop across ELPs when registered in different mapping systems. The TTL copying procedures for reencapsulating tunnels and recursive tunnels in [LISP] MUST be followed.

## 6. Service Chaining

An ELP can be used to deploy services at each reencapsulation point in the network. One example is to implement a scrubber service when a destination EID is being DoS attacked. That is, when a DoS attack is recognized when the encapsulation path is between ITR and ETR, an ELP can be registered for a destination EID to the mapping database system. The ELP can include an RTR so the ITR can encapsulate packets to the RTR which will decapsulate and deliver packets to a scrubber service device. The scrubber could decide if the offending packets are dropped or allowed to be sent to the destination EID. In which case, the scrubber delivers packets back to the RTR which encapsulates to the ETR.

## 7. RLOC Probing by RTRs

Since an RTR knows the next tunnel hop to encapsulate to, it can monitor the reachability of the next-hop RTR RLOC by doing RLOC-probing according to the procedures in [LISP]. When the RLOC is determined unreachable by the RLOC-probing mechanisms, the RTR can use another locator in the locator-set. That could be the final ETR, a RLOC of another RTR, or an ELP where it must search for itself and use the next RLOC in the ELP list to encapsulate to.

RLOC-probing can also be used to measure delay on the path between RTRs and when it is desirable switch to another lower delay ELP.

## 8. Interworking Considerations

[LISP-IW] defines procedures for how non-LISP sites talk to LISP sites. The network elements defined in the Interworking specification, the proxy ITR (PITR) and proxy ETR (PETR) (as well as their multicast counterparts defined in [LISP-MCAST]) can participate in LISP-TE. That is, a PITR and a PETR can appear in an ELP list and act as an RTR.

Note when an RLOC appears in an ELP, it can be of any address-family. There can be a mix of IPv4 and IPv6 locators present in the same ELP. This can provide benefits where islands of one address-family or the other are supported and connectivity across them is necessary. For instance, an ELP can look like:

```
(x4, a46, b64, y4, etr)
```

Where an IPv4 ITR will encapsulate using an IPv4 RLOC 'x4' and 'x4' could reach an IPv4 RLOC 'a46', but RTR 'a46' encapsulates to an IPv6 RLOC 'b64' when the network between them is IPv6-only. Then RTR 'b64' encapsulates to IPv4 RLOC 'y4' if the network between them is dual-stack.

Note that RTRs can be used for NAT-traversal scenarios [LISP-NATT] as well to reduce the state in both an xTR that resides behind a NAT and the state the NAT needs to maintain. In this case, the xTR only needs a default map-cache entry pointing to the RTR for outbound traffic and all remote ITRs can reach EIDs through the xTR behind a NAT via a single RTR (or a small set RTRs for redundancy).

RTRs have some scaling features to reduce the number of locator-set changes, the amount of state, and control packet overhead:

- o When ITRs and PITRs are using a small set of RTRs for encapsulating to "orders of magnitude" more EID-prefixes, the probability of locator-set changes are limited to the RTR RLOC changes versus the RLOC changes for the ETRs associated with the EID-prefixes if the ITRs and PITRs were directly encapsulating to the ETRs. This comes at an expense in packet stretch, but depending on RTR placement, this expense can be mitigated.
- o When RTRs are on-path between many pairwise EID flows, ITRs and PITRs can store a small number of coarse EID-prefixes.
- o RTRs can be used to help scale RLOC-probing. Instead of ITRs RLOC-probing all ETRs for each destination site it has cached, the ITRs can probe a smaller set of RTRs which in turn, probe the destination sites.

## 9. Multicast Considerations

ELPs have application in multicast environments. Just like RTRs can be used to provide connectivity across different address family islands, RTRs can help concatenate a multicast region of the network to one that does not support native multicast.

Note there are various combinations of connectivity that can be accomplished with the deployment of RTRs and ELPs:

- o Providing multicast forwarding between IPv4-only-unicast regions and IPv4-multicast regions.
- o Providing multicast forwarding between IPv6-only-unicast regions and IPv6-multicast regions.
- o Providing multicast forwarding between IPv4-only-unicast regions and IPv6-multicast regions.
- o Providing multicast forwarding between IPv6-only-unicast regions and IPv4-multicast regions.
- o Providing multicast forwarding between IPv4-multicast regions and IPv6-multicast regions.

An ITR or PITR can do a (S-EID,G) lookup into the mapping database. What can be returned is a typical locator-set that could be made up of the various RLOC addresses:

```
Multicast EID key:  (seid, G)
Locator-set:        ETR-A: priority 1, weight 25
                   ETR-B: priority 1, weight 25
                   g1:   priority 1, weight 25
                   g2:   priority 1, weight 25
```

An entry for host 'seid' sending to application group 'G'

The locator-set above can be used as a replication list. That is some RLOCs listed can be unicast RLOCs and some can be delivery group RLOCs. A unicast RLOC in this case is used to encapsulate a multicast packet originated by a multicast source EID into a unicast packet for unicast delivery on the underlying network. ETR-A could be a IPv4 unicast RLOC address and ETR-B could be a IPv6 unicast RLOC address.

A delivery group address is used when a multicast packet originated by a multicast source EID is encapsulated in a multicast packet for

multicast delivery on the underlying network. Group address 'g1' could be a IPv4 delivery group RLOC and group address 'g2' could be an IPv6 delivery group RLOC.

Flexibility for these various types of connectivity combinations can be achieved and provided by the mapping database system. And the RTR placement allows the connectivity to occur where the differences in network functionality are located.

Extending this concept by allowing ELPs in locator-sets, one could have this locator-set registered in the mapping database for (seid, G). For example:

```
Multicast EID key: (seid, G)
Locator-set:      (x, y, ETR-A):    priority 1, weight 50
                  (a, g, b, ETR-B): priority 1, weight 50
```

#### Using ELPs for multicast flows

In the above situation, an ITR would encapsulate a multicast packet originated by a multicast source EID to the RTR with unicast RLOC 'x'. Then RTR 'x' would decapsulate and unicast encapsulate to RTR 'y' ('x' or 'y' could be either IPv4 or IPv6 unicast RLOCs), which would decapsulate and unicast encapsulate to the final RLOC 'ETR-A'. The ETR 'ETR-A' would decapsulate and deliver the multicast packet natively to all the receivers joined to application group 'G' inside the LISP site.

Let's look at the ITR using the ELP (a, g, b, ETR-B). Here the encapsulation path would be the ITR unicast encapsulates to unicast RLOC 'a'. RTR 'a' multicast encapsulates to delivery group 'g'. The packet gets to all ETRs that have joined delivery group 'g' so they can deliver the multicast packet to joined receivers of application group 'G' in their sites. RTR 'b' is also joined to delivery group 'g'. Since it is in the ELP, it will be the only RTR that unicast encapsulates the multicast packet to ETR 'ETR-B'. Lastly, 'ETR-B' decapsulates and delivers the multicast packet to joined receivers to application group 'G' in its LISP site.

As one can see there are all sorts of opportunities to provide multicast connectivity across a network with non-congruent support for multicast and different address-families. One can also see how using the mapping database can allow flexible forms of delivery policy, rerouting, and congestion control management in multicast environments.

## 10. Security Considerations

When an RTR receives a LISP encapsulated packet, it can look at the outer source address to verify that RLOC is the one listed as the previous hop in the ELP list. If the outer source RLOC address appears before the RLOC which matches the outer destination RLOC address, the decapsulating RTR (or ETR if last hop), MAY choose to drop the packet.

## 11. IANA Considerations

At this time there are no requests for IANA.



## 12. References

### 12.1. Normative References

- [LISP] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-23.txt (work in progress).
- [LISP-IW] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking LISP with IPv4 and IPv6", draft-ietf-lisp-interworking-06.txt (work in progress).
- [LISP-MCAST] Farinacci, D., Meyer, D., Zwiebel, J., and S. Venaas, "LISP for Multicast Environments", draft-ietf-lisp-multicast-13.txt (work in progress).
- [LISP-MS] Fuller, V. and D. Farinacci, "LISP Map Server Interface", draft-ietf-lisp-ms-16.txt (work in progress).
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

### 12.2. Informative References

- [LISP-LCAF] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format", draft-farinacci-lisp-lcaf-09.txt (work in progress).
- [LISP-NATT] Ermagan, V., Farinacci, D., Lewis, D., Skriver, J., Maino, F., and C. White, "NAT traversal for LISP", draft-ermagan-lisp-nat-traversal-01.txt (work in progress).

progress).

## Appendix A. Acknowledgments

The authors would like to thank the following people for their ideas and comments. They are Albert Cabellos, Khalid Raza, and Vina Ermagan, and Gregg Schudel.

## Appendix B. Document Change Log

## B.1. Changes to draft-farinacci-lisp-te-01.txt

- o Posted July 2012.
- o Add the Lookup bit to allow an ELP to be a list of encapsulation and/or mapping database lookup addresses.
- o Indicate that ELPs can be used for service chaining.
- o Add text to indicate that Map-Notify messages can be sent to new RTRs in a ELP so their map-caches can be pre-populated to avoid mapping database lookup packet loss.
- o Fixes to editorial comments from Gregg.

## B.2. Changes to draft-farinacci-lisp-te-00.txt

- o Initial draft posted March 2012.

Authors' Addresses

Dino Farinacci  
cisco Systems  
Tasman Ave.  
San Jose, California  
USA

Phone: 408-718-2001  
Email: dino@cisco.com

Parantap Lahiri  
Microsoft Corporation  
Redmond, WA  
USA

Email: parantal@microsoft.com

Michael Kowal  
cisco Systems  
111 Wood Avenue South  
ISELIN, NJ  
USA

Email: mikowal@cisco.com



Network Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: January 18, 2013

V. Fuller  
D. Lewis  
V. Ermagan  
A. Jain  
cisco Systems  
July 17, 2012

LISP Delegated Database Tree  
draft-fuller-lisp-ddt-03.txt

Abstract

This draft describes the LISP Delegated Database Tree (LISP-DDT), a hierarchical, distributed database which embodies the delegation of authority to provide mappings from LISP Endpoint Identifiers (EIDs) to Routing Locators (RLOCs). It is a statically-defined distribution of the EID namespace among a set of LISP-speaking servers, called DDT nodes. Each DDT node is configured as "authoritative" for one or more EID-prefixes, along with the set of RLOCs for Map Servers or "child" DDT nodes to which more-specific EID-prefixes are delegated.

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 <http://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 January 18, 2013.

Copyright Notice

Copyright (c) 2012 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 (<http://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 . . . . .	4
2. Definition of Terms . . . . .	6
3. Database organization . . . . .	8
3.1. EID-prefix tree structure and instance IDs . . . . .	8
3.2. Configuring prefix delegation . . . . .	8
3.2.1. The root DDT node . . . . .	8
4. The Map-Referral message . . . . .	10
4.1. Action codes . . . . .	10
4.2. Referral Set . . . . .	11
4.3. Incomplete flag . . . . .	11
5. DDT network elements and their operation . . . . .	12
5.1. DDT node . . . . .	12
5.1.1. Match of a delegated prefix (or sub-prefix) . . . . .	12
5.1.2. Missing delegation from an authoritative prefix . . . . .	12
5.2. DDT Map Server . . . . .	13
5.3. DDT Map Resolver . . . . .	13
5.3.1. Queuing and sending DDT Map-Requests . . . . .	13
5.3.2. Receiving and following referrals . . . . .	14
5.3.3. Handling referral errors . . . . .	15
5.3.4. Referral loop detection . . . . .	16
6. Psuedo Code and Decision Tree diagrams . . . . .	17
6.1. Map Resolver processing of ITR Map-Request . . . . .	17
6.1.1. Pseudo-code summary . . . . .	17
6.1.2. Decision tree diagram . . . . .	18
6.2. Map Resolver processing of Map-Referral message . . . . .	19
6.2.1. Pseudo-code summary . . . . .	19
6.2.2. Decision tree diagram . . . . .	21
6.3. DDT Node processing of DDT Map-Request message . . . . .	22
6.3.1. Pseudo-code summary . . . . .	22
6.3.2. Decision tree diagram . . . . .	23
7. Example topology and request/referral following . . . . .	24
7.1. Lookup of 10.1.1.1/32 by ITR1 . . . . .	25
7.2. Lookup of 10.17.8.1/32 by ITR2 . . . . .	26
7.3. Lookup of 10.2.2.2/32 by ITR1 . . . . .	27
7.4. Lookup of 10.16.2.1/32 by ITR2 . . . . .	27
7.5. Lookup of 10.16.0.1/32 (non-existant EID) by ITR2 . . . . .	28
8. Securing the database and message exchanges . . . . .	29
8.1. XEID-prefix Delegation . . . . .	29
8.2. DDT node operation . . . . .	30



8.2.1. DDT public key revocation . . . . .	30
8.3. Map Server operation . . . . .	30
8.4. Map Resolver operation . . . . .	31
9. Open Issues and Considerations . . . . .	32
10. IANA Considerations . . . . .	33
11. Security Considerations . . . . .	34
12. References . . . . .	35
12.1. Normative References . . . . .	35
12.2. Informative References . . . . .	35
Appendix A. Acknowledgments . . . . .	36
Appendix B. Map-Referral Message Format . . . . .	37
B.1. SIG section . . . . .	39
Appendix C. Encapsulated Control Message Format . . . . .	41
Authors' Addresses . . . . .	42

## 1. Introduction

[LISP] specifies an architecture and mechanism for replacing the addresses currently used by IP with two separate name spaces: relatively static Endpoint Identifiers (EIDs), used end-to-end for terminating transport-layer associations, and Routing Locators (RLOCs), which are more dynamic, are bound to topological location, and are used for routing and forwarding through the Internet infrastructure.

LISP offers a general-purpose mechanism for mapping between EIDs and RLOCs. In organizing a database of EID to RLOC mappings, this specification extends the definition of the EID numbering space by logically prepending and appending several fields for purposes of defining the database index key: Database-ID (DBID, 16 bits), Instance identifier (IID, 32-bits), Address Family Identifier (16 bits), and EID-prefix (variable, according to AFI value). The resulting concatenation of these fields is termed an "Extended EID prefix" or XEID-prefix.

The term "Extended EID" (XEID) is also used for an individual LISP EID that is further qualified through the use of an Instance ID. See [LCAF] for further discussion of the use of Instance IDs.

The DBID is provided for possible use in case a need evolves for another, higher level in the hierarchy, to allow the creation of multiple, separate database trees.

LISP-DDT is a hierarchical distributed database which embodies the delegation of authority to provide mappings, i.e. its internal structure mirrors the hierarchical delegation of address space. It also provides delegation information to Map Resolvers, which use the information to locate EID-to-RLOC mappings. A Map Resolver which needs to locate a given mapping will follow a path through the tree-structured database, contacting, one after another, the DDT nodes along that path until it reaches the leaf DDT node(s) authoritative for the mapping it is seeking.

LISP-DDT defines a new device type, the DDT node, that is configured as authoritative for one or more XEID-prefixes. It also is configured with the set of more-specific sub-prefixes that are further delegated to other DDT nodes. To delegate a sub-prefix, the "parent" DDT node is configured with the RLOCs of each child DDT node that is authoritative for the sub-prefix. Each RLOC either points to a Map Server (sometimes termed a "terminal DDT node") to which an Egress Tunnel Routers (ETRs) has registered that sub-prefix or points to another DDT node in the database tree that further delegates the sub-prefix. See [LISP-MS] for a description of the functionality of

the Map Server and Map Resolver. Note that the target of a delegation must always be an RLOC (not an EID) to avoid any circular dependency.

To provide a mechanism for traversing the database tree, LISP-DDT defines a new LISP message type, the Map-Referral, which is returned to the sender of a Map-Request when the receiving DDT node can refer the sender to another DDT node that has more detailed information. See Section 4 for the definition of the Map-Referral message.

To find an EID-to-RLOC mapping, a LISP-DDT client, usually a DDT Map Resolver, starts by sending an Encapsulated Map-Request to a preconfigured DDT node RLOC. The DDT node responds with a Map-Referral message that either indicates that it will find the requested mapping to complete processing of the request or that the DDT client should contact another DDT node that has more-specific information; in the latter case, the DDT node then sends a new Encapsulated Map-Request to the next DDT node and the process repeats in an iterative manner.

Conceptually, this is similar to the way that a client of the Domain Name System (DNS) follows referrals (DNS responses that contain only NS records) from a series of DNS servers until it finds an answer.

## 2. Definition of Terms

**Extended EID (XEID):** a LISP EID, optionally extended with a non-zero Instance ID (IID) if the EID is intended for use in a context where it may not be a unique value, such as on a Virtual Private Network where [RFC1918] address space is used. See "Using Virtualization and Segmentation with LISP" in [LISP] for more discussion of Instance IDs.

**XEID-prefix:** a LISP EID-prefix with 16-bit LISP-DDT DBID (provided to allow the definition of multiple databases; currently always zero in this version of DDT, with other values reserved for future use), 32-bit IID and 16-bit AFI prepended. An XEID-prefix is used as a key index into the database.

**DDT node:** a network infrastructure component responsible for specific XEID-prefix and for delegation of more-specific sub-prefixes to other DDT nodes.

**DDT client:** a network infrastructure component that sends Map-Request messages and implements the iterative following of Map-Referral results. Typically, a DDT client will be a Map Resolver but it is also possible for an ITR to implement DDT client functionality.

**DDT Map Server:** a DDT node that also implements Map Server functionality (forwarding Map-Requests and/or returning Map-Replies if offering proxy Map-Reply service) for a subset of its delegated prefixes.

**DDT Map Resolver:** a network infrastructure element that accepts a Map-Request, adds the XEID to its pending request list, then queries one or more DDT nodes for the requested EID, following returned referrals until it receives one with action code MS-ACK (or an error indication). MS-ACK indicates that the Map-Request has been sent to a Map Server that will forward it to an ETR that, in turn, will provide a Map-Reply to the original sender. A DDT Map Resolver maintains both a cache of Map-Referral message results containing RLOCs for DDT nodes responsible for XEID-prefixes of interest (termed the "referral cache") a pending request list of XEIDs that are being resolved through iterative querying of DDT nodes.

**Encapsulated Map-Request:** a LISP Map-Request carried within an Encapsulated Control Message, which has an additional LISP header prepended. Sent to UDP destination port 4342. The "outer" addresses are globally-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 as documented in [LISP-MS].

**DDT Map-Request:** an Encapsulated Map-Request sent by a DDT client to a DDT node. The "DDT-originated" flag is set in the encapsulation header indicating that the DDT node should return Map-Referral messages if the Map-Request EID matches a delegated XEID-prefix known to the DDT node. Section 5.3.1 describes how DDT Map-Requests are sent.

**Authoritative XEID-prefix:** an XEID-prefix delegated to a DDT node and for which the DDT node may provide further delegations of more-specific sub-prefixes.

**Map-Referral:** a LISP message sent by a DDT node in response to a DDT Map-Request for an XEID that matches a configured XEID-prefix delegation. A non-negative Map-Referral includes a "referral", a set of RLOCs for DDT nodes that have more information about the sub-prefix; a DDT client "follows the referral" by sending another DDT Map-Request to one of those RLOCs to obtain either an answer or another referral to DDT nodes responsible for a more-specific XEID-prefix. See Section 5.1 and Section 5.3.2 for details on the sending and processing of Map-Referral messages.

**negative Map-Referral:** a Map-Referral sent in response to a DDT Map-Request that matches an authoritative XEID-prefix but for which there is no delegation configured (or no ETR registration if sent by a DDT Map-Server).

**Pending Request List:** the set of outstanding requests for which a DDT Map Resolver has received encapsulated Map-Requests from a DDT client for an XEID. Each entry in the list contains additional state needed by the referral following process, including the requestor(s) of the XEID (typically, one or more ITRs), saved information about the last referral received and followed (matching XEID-prefix, action code, RLOC set, index of last RLOC queried in the RLOC set), and any [LISP-SEC] information that was included in the DDT client Map-Request. An entry in the list may be interchangeably termed a "pending request list entry" or simply a "pending request".

For definitions of other terms, notably Map-Request, Map-Reply, Ingress Tunnel Router (ITR), Egress Tunnel Router (ETR), Map Server, and Map Resolver, please consult the LISP specification [LISP] and the LISP Mapping Service specification [LISP-MS].

### 3. Database organization

#### 3.1. EID-prefix tree structure and instance IDs

LISP-DDT defines a tree structure that is indexed by a binary encoding of five fields, in order of significance: DBID (16 bits), Instance Identifier (IID, 32 bits), Address Family Identifier (AFI, 16 bits), and EID-prefix (variable, according to AFI value). The fields are concatenated, with the most significant fields as listed above. The index into this structure is also referred to as an Extended EID-prefix (XEID-prefix).

It is important to note that LISP-DDT does not store actual EID-to-RLOC mappings; it is, rather, a distributed index that can be used to find the devices (Map Servers and their registered EIDs) that can be queried with LISP to obtain those mappings. Changes to EID-to-RLOC mappings are made on the ETRs which define them, not to any DDT node configuration. DDT node configuration changes are only required when branches of the database hierarchy are added, removed, or modified.

#### 3.2. Configuring prefix delegation

Every DDT node is configured with one or more XEID-prefixes for which it is authoritative along with a list of delegations of XEID-prefixes to other DDT nodes. A DDT node is required to maintain a list of delegations for all sub-prefixes of its authoritative XEID-prefixes; it also may list "hints", which are prefixes that it knows about that belong to its parents, to the root, or to any other point in the XEID-prefix hierarchy. A delegation (or hint) consists of an XEID-prefix, a set of RLOCs for DDT nodes that have more detailed knowledge of the XEID-prefix, and accompanying security information. Those RLOCs are returned in Map-Referral messages when the DDT node receives a DDT Map-Request with an xEID that matches a delegation. A DDT Map Server will also have a set of sub-prefixes for which it accepts ETR mapping registrations and for which it will forward (or answer, if it provides proxy Map-Reply service) Map-Requests. For details of security information in Map-Referrals see Section 8.

##### 3.2.1. The root DDT node

The root DDT node is the logical "top" of the database hierarchy: DBID=0, IID=0, AFI=0, EID-prefix=0/0. A DDT Map-Request that matches no configured XEID-prefix will be referred to the root node. The root node in a particular instantiation of LISP-DDT must therefore be configured with delegations for at least all defined IIDs and AFIs.

To aid in defining a "sub-root" DDT node that is responsible for all EID-prefixes within multiple IIDs (say, for using LISP to create

virtual networks that use overlapping address space), it may be useful to implement configuration language that allows for a range of IIDs to be delegated together. Additional configuration shorthand for delegating a range of IIDs (and all of the EIDs under them) may also be helpful.

#### 4. The Map-Referral message

A Map-Referral message is sent by a DDT node to a DDT client in response to a DDT Map-Request message. The message consists of an action code along with delegation information about the XEID-prefix that matches the XEID requested.

See Appendix B for a detailed layout of the Map-Referral message fields.

##### 4.1. Action codes

The action codes are as follows:

NODE-REFERRAL (0): indicates that the replying DDT node has delegated an XEID-prefix that matches the requested XEID to one or more other DDT nodes. The Map-Referral message contains a "map-record" with additional information, most significantly the set of RLOCs to which the prefix has been delegated, that is used by a DDT Map Resolver to "follow" the referral.

MS-REFERRAL (1): indicates that the replying DDT node has delegated an XEID-prefix that matches the requested XEID to one or more DDT Map Servers. It contains the same additional information as a NODE-REFERRAL but is handled slightly differently by the receiving DDT client (see Section 5.3.2).

MS-ACK (2): indicates that a replying DDT Map Server received a DDT Map-Request that matches an authoritative XEID-prefix for which it has one or more registered ETRs. This means that the request can be forwarded to one of those ETRs to provide an answer to the querying ITR.

MS-NOT-REGISTERED (3): indicates that the replying DDT Map Server received a Map-Request for one of its configured XEID-prefixes which has no ETRs registered.

DELEGATION-HOLE (4): indicates that the requested XEID matches a non-delegated sub-prefix of the XEID space. This is a non-LISP "hole", which has not been delegated to any DDT Map Server or ETR. See Section 5.1.2 for details.

NOT-AUTHORITATIVE (5): indicates that the replying DDT node received a Map-Request for an XEID-request for which it is not authoritative. This can occur if a cached referral has become invalid due to a change in the database hierarchy.



#### 4.2. Referral Set

For "positive" action codes (NODE-REFERRAL, MS-REFERRAL, MS-ACK), a DDT node includes in the Map-Referral message a list of RLOCs for all DDT nodes that are authoritative for the XEID-prefix being returned; a DDT Map Resolver uses this information to contact one of those DDT nodes as it "follows" a referral.

#### 4.3. Incomplete flag

A DDT node sets the "Incomplete" flag in a Map-Referral message if the Referral Set is incomplete; this is intended to prevent a DDT Map Resolver from caching a referral with incomplete information. A DDT node must set the "incomplete" flag in the following cases:

- o If it is setting action code MS-ACK or MS-NOT-REGISTERED but does not have configuration for other "peer" DDT nodes that are also authoritative for the matched XEID-prefix.
- o If it is setting action code NOT-AUTHORITATIVE.

## 5. DDT network elements and their operation

As described above, DDT introduces a new network element, the "DDT node", extends the functionality of Map Servers and Map Resolvers to send and receive Map-Referral messages. The operation of each of these devices is described as follows.

### 5.1. DDT node

When a DDT node receives a DDT Map-Request, it compares the requested XEID against its list of XEID-prefix delegations and its list of authoritative XEID-prefixes and acts as follows:

#### 5.1.1. Match of a delegated prefix (or sub-prefix)

If the requested XEID matches one of the DDT node's delegated prefixes, then a Map-Referral message is returned with the matching more-specific XEID-prefix and the set of RLOCs for the referral target DDT nodes including associated security information (see Section 8 for details on security). If the delegation is known to be a DDT Map Server, then the Map-Referral message is sent with action code MS-REFERRAL to indicate to the receiver that LISP-SEC information (if saved in the pending request) should be included in the next DDT Map-Request; otherwise, the action code NODE-REFERRAL is used.

Note that a matched delegation does not have to be for a sub-prefix of an authoritative prefix; in addition to being configured to delegate sub-prefixes of an authoritative prefix, a DDT node may also be configured with other XEID-prefixes for which it can provide referrals to DDT nodes anywhere in the database hierarchy. This capability to define "shortcut hints" is never required to be configured but may be a useful heuristic for reducing the number of iterations needed to find an EID, particular for private network deployments.

#### 5.1.2. Missing delegation from an authoritative prefix

If the requested XEID did not match a configured delegation but does match an authoritative XEID-prefix, then the DDT node returns a negative Map-Referral that uses the least-specific XEID-prefix that does not match any XEID-prefix delegated by the DDT node. The action code is set to DELEGATION-HOLE; this indicates that the XEID is not a LISP destination.

If the requested XEID did not match either a configured delegation or an authoritative XEID-prefix, then the request is dropped and a negative Map-Referral with action code NOT-AUTHORITATIVE is returned.

## 5.2. DDT Map Server

When a DDT Map Server receives a DDT Map-Request, its operation is similar to that of a DDT node with additional processing as follows:

- o If the requested XEID matches a registered XEID-prefix, then the Map-Request is forwarded to one of the destination ETR RLOCs (or the Map Server sends a Map-Reply, if it is providing proxy Map-Reply service) and a Map-Referral with the MS-ACK action is returned to the sender of the DDT Map-Request.
- o If the requested XEID matches a configured XEID-prefix for which no ETR registration has been received then a negative Map-Referral with action code MS-NOT-REGISTERED is returned to the sender of the DDT Map-Request.

## 5.3. DDT Map Resolver

Just as any other Map Resolver, a DDT Map Resolver accepts Map-Requests from its clients (typically, ITRs) and ensures that those Map-Requests are forwarded to the correct ETR, which generates Map-Replies. Unlike a Map Resolver that uses the ALT mapping system [LISP-ALT], however, a DDT Map Resolver uses an iterative process of following referrals to find the correct ETR to answer a Map-Request; this requires a DDT Map Resolver to maintain additional state: a Map-Referral cache and pending request list of XEIDs that are going through the iterative referral process.

### 5.3.1. Queuing and sending DDT Map-Requests

When a DDT Map Resolver receives an encapsulated Map-Request, it first performs a longest-match search for the XEID in its referral cache. If no match is found or if a negative cache entry is found, then the destination is not in the database; a negative Map-Reply is returned and no further processing is performed by the DDT Map Resolver.

If a match is found, the DDT Map Resolver creates a pending request list entry for the XEID and saves the original Map-Request (minus the encapsulation header) along with other information needed to track progress through the iterative referral process; the "referral XEID-prefix" is also initialized to the null value since no referral has yet been received. The Map Resolver then creates a DDT Map-Request (which is an encapsulated Map-Request with the "DDT-originated" flag set in the message header) for the XEID but without any authentication data that may have been included in the original Map-Request. It sends the DDT Map-Request to one of the RLOCs in the chosen referral cache entry. The referral cache is initially

populated with one or more statically-configured entries; additional entries are added when referrals are followed, as described below. A DDT Map Resolver is not absolutely required to cache referrals but it doing so decreases latency and reduces lookup delays.

Note that in normal use on the public Internet, the statically-configured initial referral cache for a DDT Map Resolver should include a "default" entry with RLOCs for one or more DDT nodes that can reach the DDT root node. If a Map Resolver does not have such configuration, it will return a Negative Map-Reply if it receives a query for an EID outside the subset of the mapping database known to it. While this may be desirable on private network deployments or during early transition to LISP when few sites are using it, this behavior is not appropriate when LISP is in general use on the Internet.

#### 5.3.2. Receiving and following referrals

After sending a DDT Map-Request, a DDT Map Resolver expects to receive a Map-Referral response. If none occurs within the timeout period, the DDT Map Resolver retransmits the request, sending to the next RLOC in the referral cache entry if one is available. If the maximum number of retransmissions has occurred and all RLOCs have been tried, then the pending request list entry is dequeued.

A DDT Map Resolver processes a received Map-Referral message according to each action code:

**NODE-REFERRAL:** The DDT Map Resolver checks for a possible referral loop as as described in Section 5.3.4. If no loop is found, the DDT Map Resolver saves the prefix returned in the Map-Referral message in the referral cache, updates the saved prefix and saved RLOCs in the pending request list entry, and follows the referral by sending a new DDT Map-Request to one of the DDT node RLOCs listed in the Referral Set; security information saved with the original Map-Request is not included.

**MS-REFERRAL:** The DDT Map Resolver follows an MS-REFERRAL in the same manner as a NODE-REFERRAL except that that security information saved with the original Map-Request is included in the new Map-Request sent to a Map Server (see Section 8 for details on security).

**MS-ACK:** This is returned by a DDT Map Server to indicate that it has one or more registered ETRs that can answer a Map-Request for the XEID. If the pending request did not include saved LISP-SEC information or if that information was already included in the previous DDT Map-Request (sent by the DDT Map Resolver in response

to either an MS-REFERRAL or a previous MS-ACK referral), then the pending request for the XEID is complete and is dequeued. Otherwise, LISP-SEC information is required and has not yet been sent to the authoritative DDT Map-Server; the DDT Map Resolver re-sends the DDT Map-Request with LISP-SEC information included and the pending request queue entry remains until another Map-Referral with MS-ACK action code is received. If the "incomplete" flag is not set, the prefix is saved in the referral cache.

**MS-NOT-REGISTERED:** The DDT Map Server queried could not process the request because it did not have any ETRs registered for a matching, authoritative XEID-prefix. If the DDT Map Resolver has not yet tried all of the RLOCs saved with the pending request, then it sends a Map-Request to the next RLOC in that list. If all RLOCs have been tried, then the destination XEID is not registered and is unreachable. The DDT Map Resolver returns a negative Map-Reply to the original Map-Request sender; this Map-Reply contains the non-registered XEID-prefix with TTL value of one minute. A negative referral cache entry is created for the prefix (also with TTL of one minute) and the pending request is dequeued.

**DELEGATION-HOLE:** The DDT Map Server queried did not have an XEID-prefix defined that matched the requested XEID so it does not exist in the mapping database. The DDT Map Resolver returns a negative Map-Reply to the original Map-Request sender; this Map-Reply will indicate the least-specific XEID-prefix matching the requested XEID for which no delegations exist and will have a TTL value of 15 minutes. A negative referral cache entry is created for the prefix (also with TTL of 15 minutes) and the pending request is dequeued.

**NOT-AUTHORITATIVE:** The DDT Map Server queried is not authoritative for the requested XEID. This can occur if a cached referral has become invalid due to a change in the database hierarchy. If the DDT Map Resolver receiving this message can determine that it is using old cached information, it may choose to delete that cached information and re-try the original Map-Request, starting from its "root" cache entry. If this action code is received in response to a query that was not to cached referral information, then it indicates a database synchronization problem or configuration error. The pending request list entry that caused this answer is removed, with no answer returned to the original requestor.

### 5.3.3. Handling referral errors

Other states are possible, such as a misconfigured DDT node (acting as a proxy Map Server, for example) returning a Map-Reply to the DDT Map Resolver; they should be considered errors and logged as such.

It is not clear exactly what else the DDT Map Resolver should do in such cases; one possibility is to remove the pending request list entry and send a negative Map-Reply to the original Map-Request sender. Alternatively, if a DDT Map Resolver detects unexpected behavior by a DDT node, it could mark that node as unusable in its referral cache and update the pending request to try a different DDT node if more than one is listed in the referral cache. In any case, any prefix contained in a Map-Referral message that causes a referral error (including a referral loop) is not saved in the DDT Map-Resolver referral cache.

#### 5.3.4. Referral loop detection

In response to a Map-Referral message with action code NODE-REFERRAL or MS-REFERRAL, a DDT Map Resolver is directed to query a new set of DDT node RLOCs that are expected to have more-specific XEID-prefix information for the requested XEID. To prevent a possible "iteration loop" (following referrals back-and-forth among a set of DDT nodes without ever finding an answer), a DDT Map Resolver saves the last received referral XEID-prefix for each pending request and checks that a newly received NODE-REFERRAL or MS-REFERRAL message contains a more-specific referral XEID-prefix; an exact or less-specific match of the saved XEID-prefix indicates a referral loop. If a loop is detected, the DDT Map Resolver handles the request as described in Section 5.3.3. Otherwise, the Map Resolver saves the most recently received referral XEID-prefix with the pending request when it follows the referral.

As an extra measure to prevent referral loops, it is probably also wise to limit the total number of referrals for any request to some reasonable number; the exact value of that number will be determined during experimental deployment of LISP-DDT but is bounded by the maximum length of the XEID.

Note that when a DDT Map Resolver adds an entry to its lookup queue and sends an initial Map-Request for an XEID, the queue entry has no previous referral XEID-prefix; this means that the first DDT node contacted by a DDT Map Resolver may provide a referral to anywhere in the DDT hierarchy. This, in turn, allows a DDT Map Resolver to use essentially any DDT node RLOCs for its initial cache entries and depend on the initial referral to provide a good starting point for Map-Requests; there is no need to configure the same set of root DDT nodes on all DDT Map Resolvers.

## 6. Psuedo Code and Decision Tree diagrams

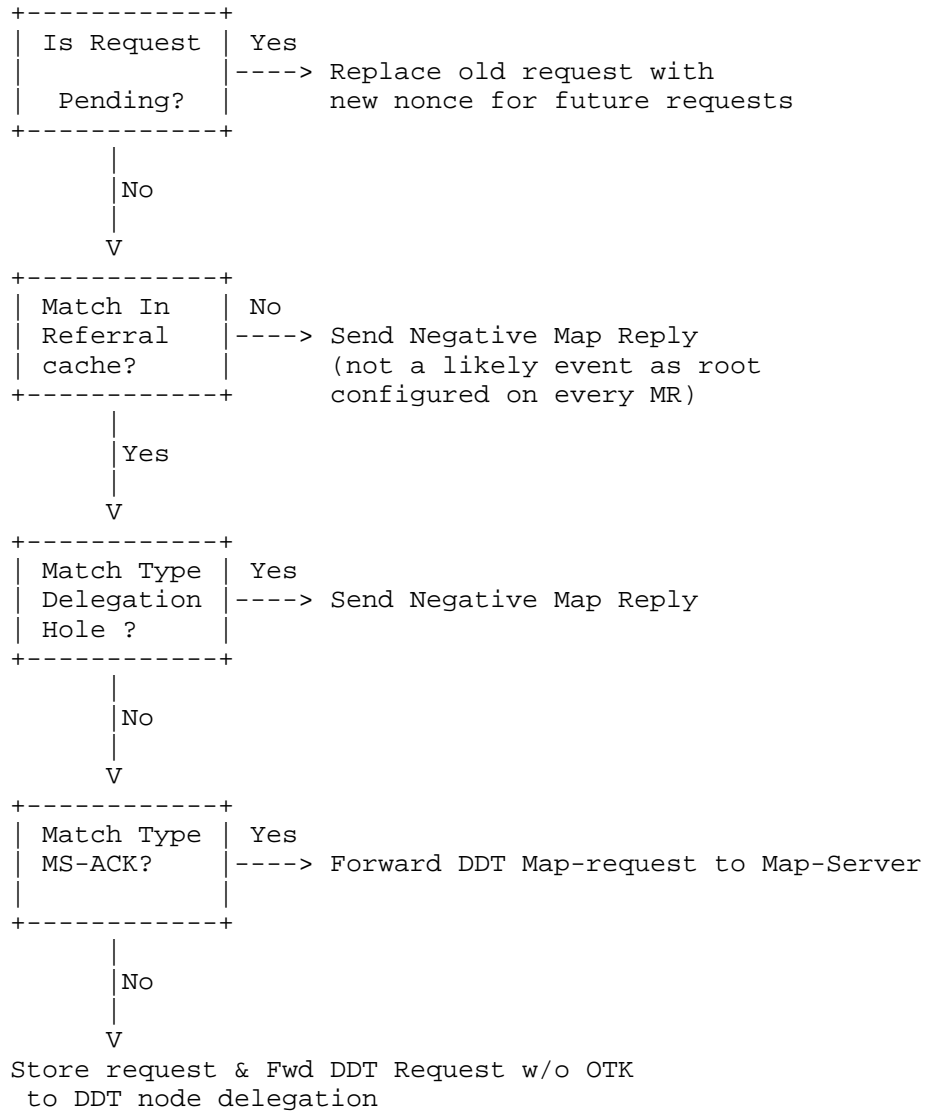
To aid in implementation, each of the major DDT Map Server and DDT Map Resolver functions are described below, first using simple "psuedo-code" and then in the form of a decision tree.

### 6.1. Map Resolver processing of ITR Map-Request

#### 6.1.1. Pseudo-code summary

```
if ( request pending i.e., (ITR,EID) of request same ) {  
    replace old request with new & use new request nonce  
    for future requests  
} else if ( no match in refcache ) {  
    return negative map-reply to ITR  
} else if ( match type delegation hole ) {  
    return negative map-reply to ITR  
} else if ( match type ms-ack ) {  
    fwd DDT request to map-server  
} else {  
    store & fwd DDT request w/o OTK to node delegation  
}
```

## 6.1.2. Decision tree diagram





## 6.2. Map Resolver processing of Map-Referral message

### 6.2.1. Pseudo-code summary

```
if ( no request pending matched by referral nonce ) {
    silently drop
}

if ( pfx in referral less specific than last referral used ) {
    if ( gone through root ) {
        silently drop
    } else {
        goto root
    }
}

switch (map_referral_type) {

    case NOT_AUTHORITATIVE :
        if ( gone through root ) {
            return negative map-reply to ITR
        } else {
            goto root
        }

    case DELEGATION_HOLE:
        cache & send negative map-reply to ITR

    case MS_REFERRAL:
        if ( referral equal to last used ) {
            if ( gone through root ) {
                return negative map-reply to ITR
            } else {
                goto root
            }
        } else {
            cache & follow the referral
        }

    case NODE_REFERRAL:
        if ( referral equal to last used ) {
            if ( gone through root ) {
                return negative map-reply to ITR
            } else {
                goto root
            }
        } else {
            cache & follow the referral
        }
}
```

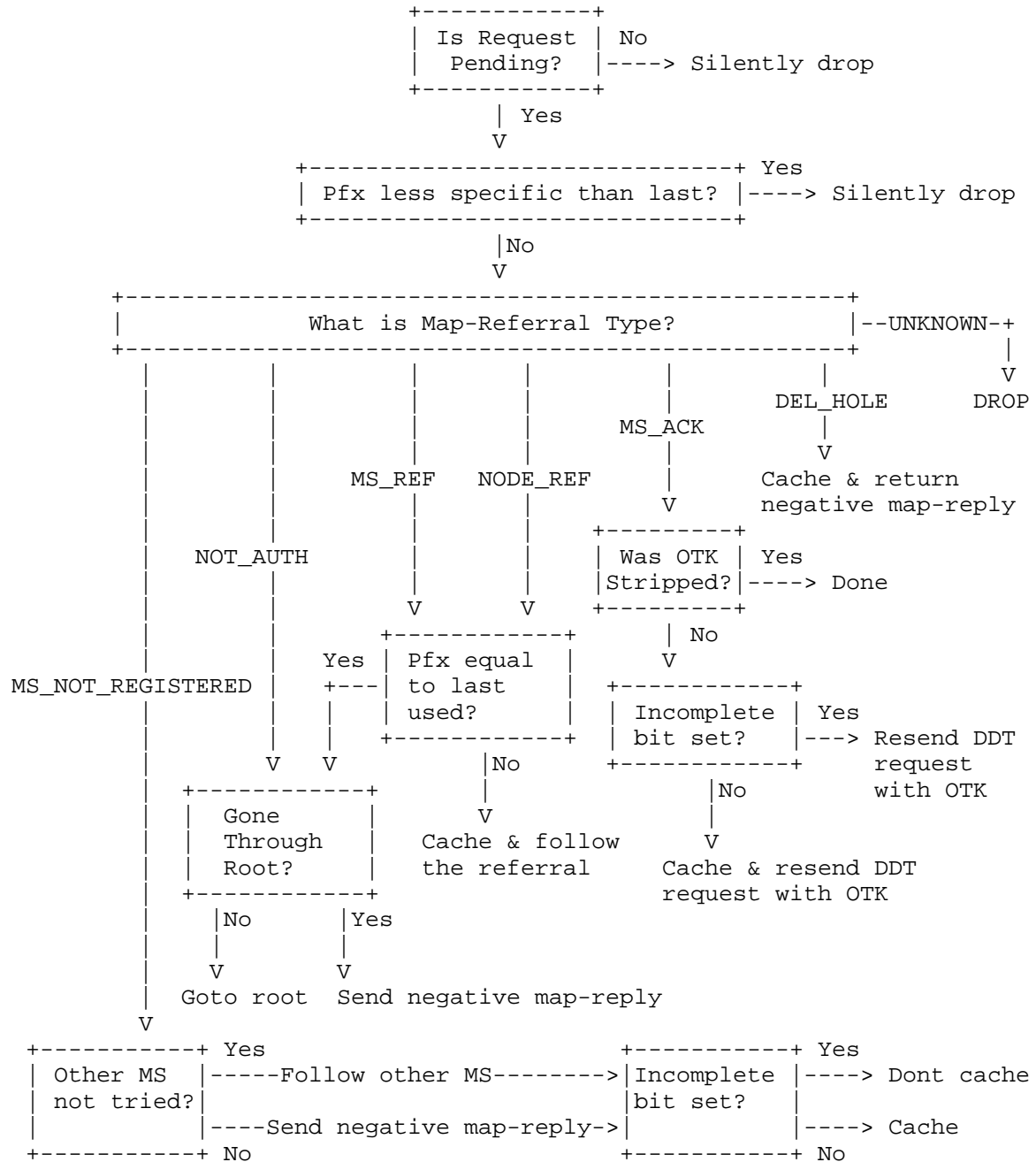
```
    }

    case MS_ACKNOWLEDGEMENT:
        if ( OTK stripped ) {
            if ( incomplete ) {
                resend request with OTK
            } else {
                cache & resend request with OTK
            }
        }

    case MS_NOT_REGISTERED:
        if { all map-server delegations not tried } {
            follow delegations not tried
            if ( !incomplete ) {
                cache
            }
        } else {
            send negative map-reply to ITR
            if { !incomplete } {
                cache
            }
        }
    }

    case DEFAULT:
        drop
    }
}
```

## 6.2.2. Decision tree diagram

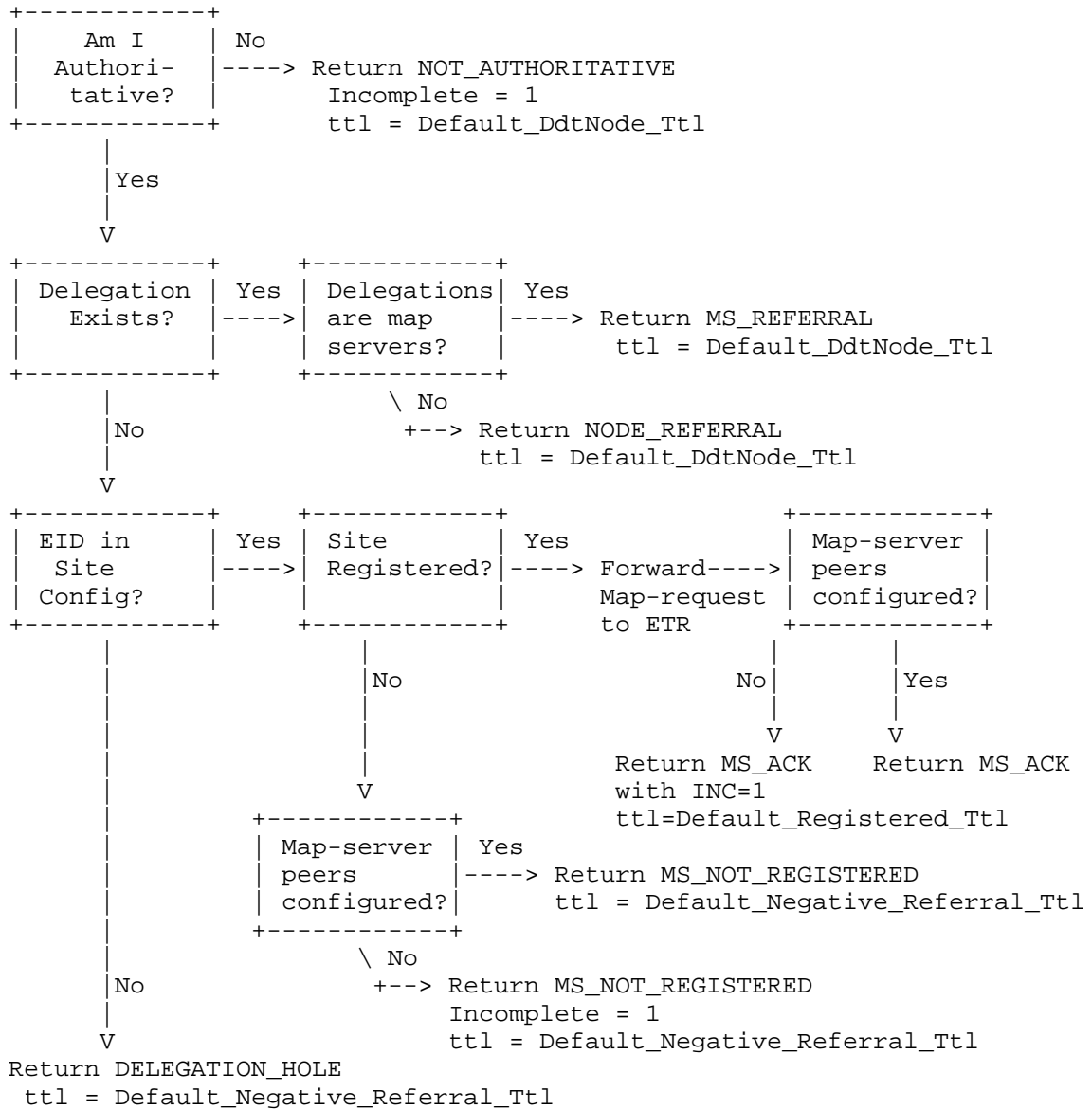


## 6.3. DDT Node processing of DDT Map-Request message

## 6.3.1. Pseudo-code summary

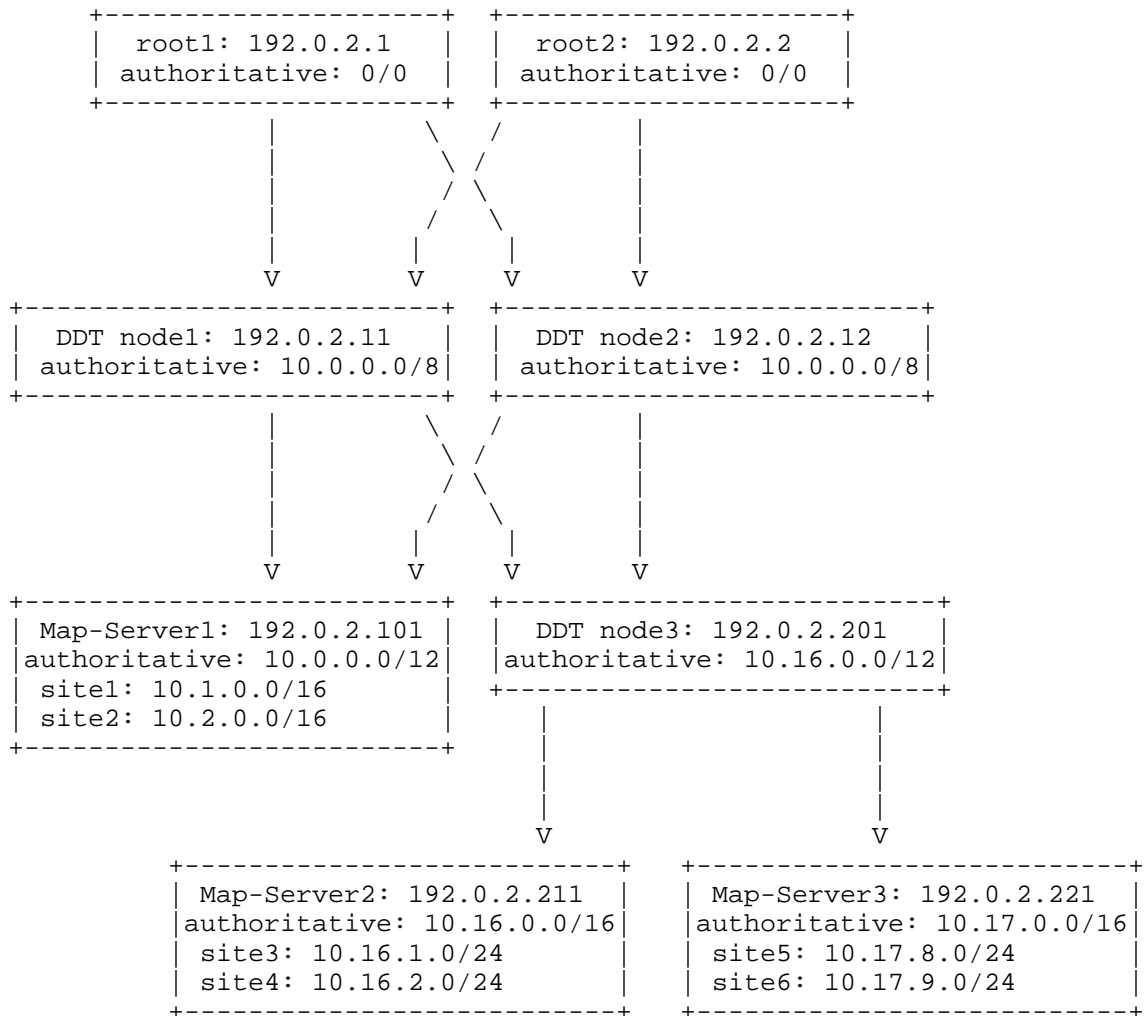
```
if ( I am not authoritative ) {
    send map-referral NOT_AUTHORITATIVE with
        incomplete bit set and ttl 0
} else if ( delegation exists ) {
    if ( delegated map-servers ) {
        send map-referral MS_REFERRAL with
            ttl 'Default_DdtNode_Ttl'
    } else {
        send map-referral NODE_REFERRAL with
            ttl 'Default_DdtNode_Ttl'
    }
} else {
    if ( eid in site ) {
        if ( site registered ) {
            forward map-request to etr
            if ( map-server peers configured ) {
                send map-referral MS_ACKNOWLEDGEMENT with
                    ttl 'Default_Registered_Ttl'
            } else {
                send map-referral MS_ACKNOWLEDGEMENT with
                    ttl 'Default_Registered_Ttl' and incomplete bit set
            }
        } else {
            if ( map-server peers configured ) {
                send map-referral MS_NOT_REGISTERED with
                    ttl 'Default_Configured_Not_Registered_Ttl'
            } else {
                send map-referral MS_NOT_REGISTERED with
                    ttl 'Default_Configured_Not_Registered_Ttl'
                    and incomplete bit set
            }
        }
    } else {
        send map-referral DELEGATION_HOLE with
            ttl 'Default_Negative_Referral_Ttl'
    }
}
```

## 6.3.2. Decision tree diagram



## 7. Example topology and request/referral following

To show how referrals are followed to find the RLOCs for a number of EIDs, consider the following example EID topology for DBID=0, IID=0, AFI=1, and EID=0/0



DDT nodes are configured for this "root" at IP addresses 192.0.2.1 and 192.0.2.2. DDT Map Resolvers are configured with default referral cache entries to these addresses.

The root DDT nodes delegate 10.0.0/8 to two DDT nodes with IP addresses 192.0.2.11 and 192.0.2.12.

The DDT nodes for 10.0.0.0/8 delegate 10.0.0.0/12 to a DDT Map Server with RLOC 192.0.2.101

The DDT Map Server for 10.0.0.0/12 is configured to allow ETRs to register the sub-prefixes 10.1.0.0/16 and 10.2.0.0/16

The DDT nodes for 10.0.0.0/8 also delegate 10.16.0.0/12 to a DDT node with RLOC 192.0.2.201

The DDT node for 10.16.0.0/12 is further configured to delegate 10.16.0.0/16 to a DDT Map Server with RLOC 192.0.2.211 and 10.17.0.0/16 to a DDT Map Server with RLOC 192.0.2.221

The DDT Map Server for 10.16.0.0/16 is configured to allow ETRs to register the sub-prefixes 10.16.1.0/24 and 10.16.2.0/24

The DDT Map Server for 10.17.0.0/16 is configured to allow ETRs to register the sub-prefixes 10.17.8.0/24 and 10.17.9.0/24

#### 7.1. Lookup of 10.1.1.1/32 by ITR1

The first example shows a DDT Map Resolver following a delegation from the root to a DDT node followed by another delegation to a DDT Map Server.

ITR1 sends an Encapsulated Map-Request for 10.1.1.1 to one of its configured (DDT) Map Resolvers. The DDT Map Resolver proceeds as follows:

1. Send DDT Map-Request (for 10.1.1.1) to one of the root DDT nodes, 192.0.2.1 or 192.0.2.2
2. Receive (and save in referral cache) Map-Referral for EID-prefix 10.0.0.0/8, action code NODE-REFERRAL, RLOC set (192.0.2.11, 192.0.2.12)
3. Send DDT Map-Request to 192.0.2.11 or 192.0.2.12
4. Receive (and save in referral cache) Map-Referral for EID-prefix 10.0.0.0/12, action code MS-REFERRAL, RLOC set (192.0.2.101)

5. Send DDT Map-Request to 192.0.2.101; if the ITR-originated Encapsulated Map-Request had a LISP-SEC signature, it is included
6. DDT Map Server at 192.0.2.101 decapsulates the DDT Map-Request and forwards to a registered sitel ETR for 10.1.0.0/16
7. DDT Map Server at 192.0.2.101 sends a Map-Referral message for EID-prefix 10.1.0.0/16, action code MS-ACK to the DDT Map Resolver
8. DDT Map Resolver receives Map-Referral message and dequeues the pending request for 10.1.1.1
9. sitel ETR for 10.1.0.0/16 receives Map-Request forwarded by DDT Map Server and sends Map-Reply to ITR1

#### 7.2. Lookup of 10.17.8.1/32 by ITR2

The next example shows a three-level delegation: root to first DDT node, first DDT node to second DDT node, second DDT node to DDT Map Server.

ITR2 sends an Encapsulated Map-Request for 10.17.8.1 to one of its configured (DDT) Map Resolvers, which are different from those for ITR1. The DDT Map Resolver proceeds as follows:

1. Send DDT Map-Request (for 10.17.8.1) to one of the root DDT nodes, 192.0.2.1 or 192.0.2.2
2. Receive (and save in referral cache) Map-Referral for EID-prefix 10.0.0.0/8, action code NODE-REFERRAL, RLOC set (192.0.2.11, 192.0.2.12)
3. Send DDT Map-Request to 192.0.2.11 or 192.0.2.12
4. Receive (and save in referral cache) Map-Referral for EID-prefix 10.16.0.0/12, action code NODE-REFERRAL, RLOC set (192.0.2.201)
5. Send DDT Map-Request to 192.0.2.201
6. Receive (and save in referral cache) Map-Referral for EID-prefix 10.17.0.0/16, action code MS-REFERRAL, RLOC set (192.0.2.221)
7. Send DDT Map-Request to 192.0.2.221; if the ITR-originated Encapsulated Map-Request had a LISP-SEC signature, it is included



8. DDT Map Server at 192.0.2.221 decapsulates the DDT Map-Request and forwards to a registered site5 ETR for 10.17.8.0/24
9. DDT Map Server at 192.0.2.221 sends a Map-Referral message for EID-prefix 10.17.8.0/24, action code MS-ACK, to the DDT Map Resolver
10. DDT Map Resolver receives Map-Referral(MS-ACK) message and dequeues the pending request for 10.17.8.1
11. site5 ETR for 10.17.8.0/24 receives Map-Request forwarded by DDT Map Server and sends Map-Reply to ITR2

#### 7.3. Lookup of 10.2.2.2/32 by ITR1

This example shows how a DDT Map Resolver uses a saved referral cache entry to skip the referral process and go directly to a DDT Map Server for a prefix that is similar to one previously requested.

In this case, ITR1 uses the same Map Resolver used in example Section 7.1. It sends an Encapsulated Map-Request for 10.2.2.2 to that (DDT) Map Resolver. The DDT Map-Resolver finds an MS-REFERRAL cache entry for 10.0.0.0/12 with RLOC set (192.0.2.101) and proceeds as follows:

1. Send DDT Map-Request (for 10.2.2.2) to 192.0.2.101; if the ITR-originated Encapsulated Map-Request had a LISP-SEC signature, it is included
2. DDT Map Server at 192.0.2.101 decapsulates the DDT Map-Request and forwards to a registered site2 ETR for 10.2.0.0/16
3. DDT Map Server at 192.0.2.101 sends a Map-Referral message for EID-prefix 10.2.0.0/16, action code MS-ACK to the DDT Map Resolver
4. DDT Map Resolver receives Map-Referral(MS-ACK) and dequeues the pending request for 10.2.2.2
5. site2 ETR for 10.2.0.0/16 receives Map-Request and sends Map-Reply to ITR1

#### 7.4. Lookup of 10.16.2.1/32 by ITR2

This example shows how a DDT Map Resolver uses a saved referral cache entry to start the referral process at a non-root, intermediate DDT node for a prefix that is similar to one previously requested.

In this case, ITR2 asks the same Map Resolver used in example Section 7.2. It sends an Encapsulated Map-Request for 10.16.2.1 to that (DDT) Map Resolver, which finds a NODE-REFERRAL cache entry for 10.16.0.0/12 with RLOC set (192.0.2.201). It proceeds as follows:

1. Send DDT Map-Request (for 10.16.2.1) to 192.0.2.201
  2. Receive (and save in referral cache) Map-Referral for EID-prefix 10.16.0.0/16, action code MS-REFERRAL, RLOC set (192.0.2.211)
  3. Send DDT Map-Request to 192.0.2.211; if the ITR-originated Encapsulated Map-Request had a LISP-SEC signature, it is included
  4. DDT Map Server at 192.0.2.211 decapsulates the DDT Map-Request and forwards to a registered site4 ETR for 10.16.2.0/24
  5. DDT Map Server at 192.0.2.211 sends a Map-Referral message for EID-prefix 10.16.2.0/24, action code MS-ACK to the DDT Map Resolver
  6. DDT Map Resolver receives Map-Referral(MS-ACK) and dequeues the pending request for 10.16.2.1
  7. site4 ETR for 10.16.2.0/24 receives Map-Request and sends Map-Reply to ITR2
- 7.5. Lookup of 10.16.0.1/32 (non-existent EID) by ITR2

This example uses the cached MS-REFERRAL for 10.16.0.0/16 learned above to start the lookup process at the DDT Map-Server at 192.0.2.211. The DDT Map Resolver proceeds as follows:

1. Send DDT Map-Request (for 10.16.0.1) to 192.0.2.211; if the ITR-originated Encapsulated Map-Request had a LISP-SEC signature, it is included
2. DDT Map Server at 192.0.2.211, which is authoritative for 10.16.0.0/16, does not have a matching delegation for 10.16.0.1. It responds with a Map-Referral message for 10.16.0.0/24, action code DELEGATION-HOLE to the DDT Map Resolver. The prefix 10.16.0.0/24 is used because it is the least-specific prefix that does match the requested EID but does not match one of configured delegations (10.16.1.0/24 and 10.16.2.0/24).
3. DDT Map Resolver receives the delegation, adds a negative referral cache entry for 10.16.0.0/24, dequeues the pending request for 10.16.0.1, and returns a negative Map-Reply to ITR2.

## 8. Securing the database and message exchanges

This section specifies the DDT security architecture that provides data origin authentication, data integrity protection, and XEID-prefix delegation. Global XEID-prefix authorization is out of the scope of this document.

Each DDT node is configured with one or more public/private key pair(s) that are used to digitally sign referral records for XEID-prefix(es) that the DDT node is authoritative for. In other words, each public/private key pair is associated with the combination of a DDT node and the XEID-prefix that it is authoritative for. Every DDT node is also configured with the public keys of its children DDT nodes. By including public keys of target child DDT nodes in the Map-Referral records, and signing each record with the DDT node's private key, a DDT node can securely delegate sub-prefixes of its authoritative XEID-prefixes to its children DDT nodes.

Map Resolvers are configured with one or more trusted public keys referred to as trust anchors. Trust anchors are used to authenticate the DDT security infrastructure. Map Resolvers can discover a DDT node's public key either by having it configured as a trust anchor, or by obtaining it from the node's parent as part of a signed Map-Referral. When a public key is obtained from a node's parent, it is considered trusted if it is signed by a trust anchor, or if it is signed by a key that was previously trusted. Typically, in a Map Resolver, the root DDT node public keys should be configured as trust anchors. Once a Map Resolver authenticates a public key it locally caches the key along with the associated DDT node RLOC and XEID-prefix for future use.

### 8.1. XEID-prefix Delegation

In order to delegate XEID sub-prefixes to its children, a parent DDT node signs its Map-Referrals. Every signed Map-Referral also includes the public keys associated with each child DDT node. Such a signature indicates that the parent node is delegating the specified XEID -prefix to a given child DDT node. The signature is also authenticating the public keys associated with the children nodes, and authorizing them to be used by the children DDT nodes to provide origin authentication and integrity protection for further delegations and mapping information of the XEID-prefix allocated to the DDT node.

As a result, for a given XEID-prefix, a Map Resolver can form an authentication chain from a configured trust anchor (typically the root DDT node) to the leaf nodes (Map Servers). Map Resolvers leverage this authentication chain to verify the Map-Referral

signatures while walking the DDT tree until they reach a Map Server authoritative for the given XEID-prefix.

## 8.2. DDT node operation

Upon receiving a Map-Request, the DDT node responds with a Map-Referral as specified in Section 5. For every record present in the Map-Referral, the DDT node also includes the public keys associated with the record's XEID-prefix and the RLOCs of the children DDT nodes. Each record contained in the Map-Referral is signed using the DDT node's private key.

### 8.2.1. DDT public key revocation

The node that owns a public key can also revoke that public key. For instance if a parent node advertises a public key for one of its child DDT nodes, the child DDT node can at a later time revoke that key. Since DDT nodes do not keep track of the Map Resolvers that query them, revocation is done in a pull model, where the Map Resolver is informed of the revocation of a key only when it queries the node that owns that key. If the parent DDT is configured to advertise this key, the parent node must also be signaled to remove the key from the records it advertises for the child DDT node; this is necessary to avoid further distribution of the revoked key.

To securely revoke a key, the DDT node creates a new Record for the associated XEID-prefix and locator, including the revoked key with the R bit set. The DDT node must also include a signature in the Record that covers this record; this is computed using the private key corresponding to the key being revoked. Such a record is termed a "revocation record". By including this record in its Map-Referrals, the DDT node informs querying Map Resolvers about the revoked key. A digital signature computed with a revoked key can only be used to authenticate the revocation, and should not be used to validate any data. To prevent a compromised key from revoking other valid keys, a given key can only be used to sign a revocation for that specific key; it cannot be used to revoke other keys. This prevents the use of a compromised key to revoke other valid keys as described in [RFC5011]. A revocation record must be advertised for a period of time equal to or greater than the TTL value of the Record that initially advertised the key, starting from the time that the advertisement of the key was stopped by removal from the parent DDT node.

## 8.3. Map Server operation

Similar to a DDT node, a Map Server is configured with one (or more) public/private key pairs that it must use to sign Map-Referrals.

However unlike DDT nodes, Map Servers do not delegate prefixes and as a result they do not need to include keys in the Map-Referrals they generate.

#### 8.4. Map Resolver operation

Upon receiving a Map-Referral, the Map Resolver must first verify the signature(s) by using a trust anchor, or a previously authenticated public key, associated with the DDT node sending the Map-Referral. If multiple authenticated keys are associated with the DDT node sending this Map-Referral, the Key Tag field of the signature can be used to select the right public key for verifying the signature. If the key tag matches more than one key associated with that DDT node, the Map Resolver must try verifying the signature with all matching keys. For every matching key that is found the Map Resolver must also verify that the key is authoritative for the XEID-prefix in the Map-Referral record. If such a key is found, the Map Resolver must use it to verify the associated signature in the record. If no matching key is found, or if none of the matching keys is authoritative for the XEID-prefix in the Map-Referral record, or if such a key is found but the signature is not valid the Map-Referral record is considered corrupted and must be discarded. This may be due to expired keys. The Map Resolver can try other siblings of this node if there is an alternative node authoritative for the same prefix. If not, the Map Resolver can query the DDT node's parent to retrieve a valid key. It is good practice to use a counter or timer to avoid repeating this process if the resolver cannot verify the signature after several trials.

Once the signature is verified, the Map Resolver has verified the XEID-prefix delegation in the Map-Referral, and authenticated the public keys of the children DDT nodes. The Map Resolver must add these keys to the authenticated keys associated with each child DDT node and XEID-prefix. These keys are considered valid for the duration specified in the record's TTL field.

## 9. Open Issues and Considerations

There are a number of issues with the organization of the mapping database that need further investigation. Among these are:

- o Unlike in [LISP-ALT], DDT does not currently define a mechanism for propagating ETR-to-Map Server registration state. This requires DDT Map Servers to suppress returning negative Map-Reply messages for defined but unregistered XEID-prefixes to avoid loss of connectivity during partial ETR registration failures. Suppressing these messages may cause a delay for an ITR obtaining a mapping entry when such a failure is occurring.
- o Defining an interface to implement interconnection and/or interoperability with other mapping databases, such as LISP+ALT.
- o Additional key structures for use with LISP-DDT, such as to support additional EID formats as defined in [LCAF].
- o Authentication of delegations between DDT nodes.
- o Possibility of a new, more general format for the Map-Referral messages to facilitate the use of LISP-DDT with additional DBID/IID/EID combinations. Currently-defined packet formats should be considered to be preliminary and provisional until this issue has received greater attention.
- o Management of the DDT Map Resolver referral cache, in particular, detecting and removing outdated entries.

The authors expect that experimentation on the LISP pilot network will help answer open questions surrounding these and other issues.

## 10. IANA Considerations

This document makes no request of the IANA.

## 11. Security Considerations

Section 8 describes a DDT security architecture that provides data origin authentication, data integrity protection, and XEID-prefix delegation within the DDT Infrastructure.

Global XEID-prefix authorization is beyond the scope of this document, but the SIDR working group [RFC6480] is developing an infrastructure to support improved security of Internet routing. Further work is required to determine if SIDR's public key infrastructure (PKI) and the distributed repository system it uses for storing and disseminating PKI data objects may also be used by DDT devices to verifiably assert that they are the legitimate holders of a set of XEID prefixes.

DDT security and [LISP-SEC] complement each other in securing the DDT infrastructure, Map-Referral messages and the Map-Request/Map-Reply protocol. In addition LISP-SEC can use the DDT public key infrastructure to secure the transport of LISP-SEC key material (the One-Time Key) from a Map-Resolver to the corresponding Map-Server. For this reason, when LISP-SEC is deployed in conjunction with a LISP-DDT mapping database and the path between Map-Resolver and Map-Server needs to be protected, DDT security should be enabled as well.



## 12. References

### 12.1. Normative References

- [LCAF] Farinacci, D. and J. Snijders, "LISP Canonical Address Format", draft-ietf-lisp-lcaf-06.txt (work in progress), October 2011.
- [LISP] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-22.txt (work in progress), February 2012.
- [LISP-MS] Fuller, V. and D. Farinacci, "LISP Map Server Interface", draft-ietf-lisp-ms-16.txt (work in progress), February 2012.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.
- [RFC4634] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, July 2006.
- [RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", RFC 5011, September 2007.

### 12.2. Informative References

- [LISP-ALT] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "LISP Alternative Topology (LISP-ALT)", draft-ietf-lisp-alt-10.txt (work in progress), December 2011.
- [LISP-SEC] Maino, F., Ermagan, V., Cabellos, A., Sanchez, D., and O. Bonaventure, "LISP-Security", draft-ietf-lisp-sec-01.txt (work in progress), January 2012.
- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.
- [RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, February 2012.

## Appendix A. Acknowledgments

The authors wish to express their thanks to Damien Saucez, Lorand Jakab, and Olivier Bonaventure for work on LISP-TREE and LISP iterable mappings that inspired the hierarchical database structure and lookup iteration approach described in this document. Special thanks also go to Dino Farinacci and Isidor Kouvelas for their implementation work, to Selina Heimlich and Srin Subramanian for testing, to Fabio Maino for work on security processing, and to Jesper Skriver, Andrew Partan, and Noel Chiappa; all of these individuals have participated in (and put up with) seemingly endless hours of discussion of mapping database ideas, concepts, and issues.

## Appendix B. Map-Referral Message Format

	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								
	+-----																																							

ACT: The "action" field of the mapping record in a Map-Referral message encodes 6 action types. The values for the action types are:

NODE-REFERRAL (0): Sent by a DDT node with a child delegation which is authoritative for the EID.

MS-REFERRAL (1): Sent by a DDT node that has information about Map Server(s) for the EID but it is not one of the Map Servers listed, i.e. the DDT-Node sending the referral is not a Map Server.

MS-ACK (2): Sent by a DDT Map Server that has one or more ETR registered for the EID.

MS-NOT-REGISTERED (3): Sent by a DDT Map Server that is configured for the EID-prefix but for which no ETRs are registered.

DELEGATION-HOLE (4): Sent by an intermediate DDT node with authoritative configuration covering the requested EID but without any child delegation for the EID. Also sent by a DDT Map Server with authoritative configuration covering the requested EID but for which no specific site ETR is configured.

NOT-AUTHORITATIVE (5): Sent by a DDT node that does not have authoritative configuration for the requested EID. The EID-prefix returned MUST be the original requested EID and the TTL MUST be set to 0. However, if such a DDT node has a child delegation covering the requested EID, it may choose to return NODE-REFERRAL or MS-REFERRAL as appropriate. A DDT Map Server with site information may choose to return of type MS-ACK or MS-NOT-REGISTERED as appropriate.

Incomplete: The "I" bit indicates that a DDT node's referral-set of locators is incomplete and the receiver of this message should not cache the referral. A DDT sets the "incomplete" flag, the TTL, and the Action Type field as follows:

Type (Action field)		Incomplete Referral-set		TTL values
0	NODE-REFERRAL	NO	YES	1440
1	MS-REFERRAL	NO	YES	1440
2	MS-ACK	*	*	1440
3	MS-NOT-REGISTERED	*	*	1
4	DELEGATION-HOLE	NO	NO	15
5	NOT-AUTHORITATIVE	YES	NO	0

\*: The "Incomplete" flag setting on Map Server originated referral of MS-REFERRAL and MS-NOT-REGISTERED types depend on whether the Map Server has the full peer Map Server configuration for the same prefix and has encoded the information in the mapping record. Incomplete bit is not set when the Map Server has encoded the information, which means the referral-set includes all the RLOCs of all Map Servers that serve the prefix. It is set when the Map Server has not encoded the Map Server set information.

SigCnt: Indicates the number of signatures (sig section) present in the Record. If SigCnt is larger than 0, the signature information captured in a sig section as described in Appendix B.1 will be appended to the end of the record. The number of sig sections at the end of the Record must match the SigCnt.

Loc/LCAF-AFI: If this is a Loc AFI, keys are not included in the record. If this is a LCAF AFI, the contents of the LCAF depend on

the Type field of the LCAF. Security material are stored in LCAF Type 11. DDT nodes and Map Servers can use this LCAF Type to include public keys associated with their Child DDT nodes for a XEID-prefix referral record. LCAF types and formats are defined in [LCAF].

All the field descriptions are equivalent to those in the Map-Reply message, as defined in [LISP]. Note, though, that the set of RLOCs correspond to the DDT node to be queried as a result of the referral not the RLOCs for an actual EID-to-RLOC mapping.

#### B.1. SIG section

If SigCnt field in the Map-Referral is not 0, the signature information is included at the end of captured in a sig section as described below. SigCnt counts the number of sig sections that appear at the end of the Record.

```

      +-----+
      /|                                     |
      / +-----+ Original Record TTL      |
      / |                                     |
      | +-----+ Signature Expiration      |
      | |                                     |
      | +-----+ Signature Inception      |
      | |                                     |
      | +-----+ Key Tag | Sig Length      |
      | +-----+         |                 |
      \ | Sig-Algorithm | Reserved |         |
      \ +-----+         |                 |
      \ ~               | Signature         | ~
      +-----+

```

Original Record TTL: The original Record TTL for this record that is covered by the signature. Record TTL is in minutes.

Key Tag: An identifier to specify which key is used for this signature if more than one valid key exists for the signing DDT node.

Sig Length: The length of the Signature field.

Sig-Algorithm: The identifier of the cryptographic algorithm used for the signature. Default value is RSA-SHA1.

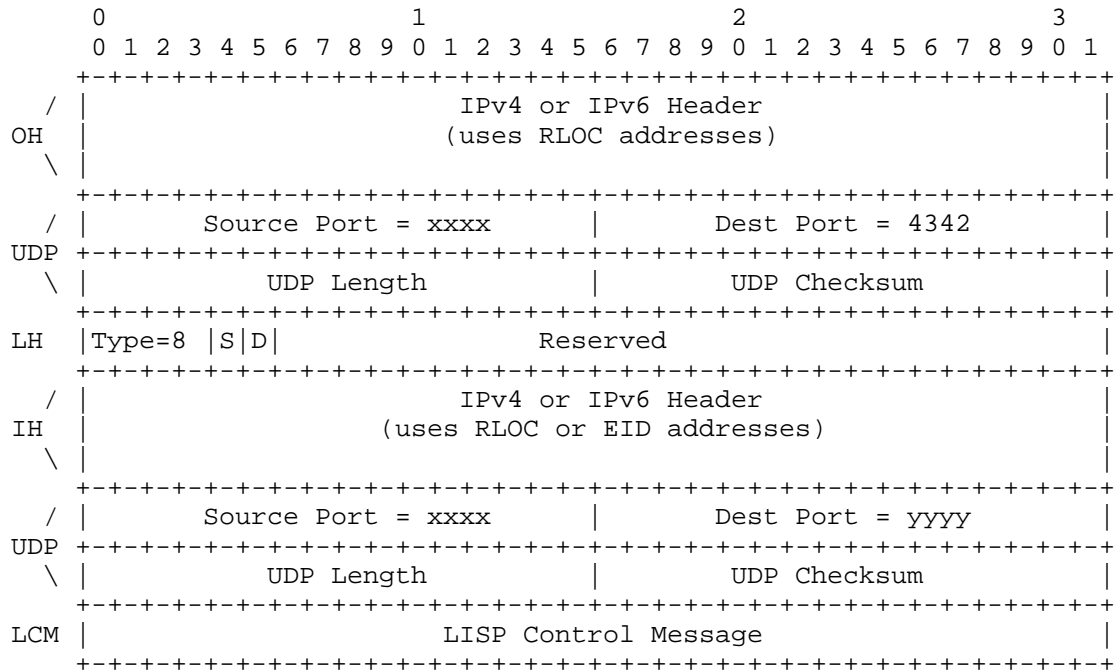
Reserved: This field must be set to 0 on transmit and must be ignored on receipt.

Signature Expiration and Inception: Specify the validity period for the signature. Each specify a date and time in the form of a 32-bit

unsigned number of seconds elapsed since 1 January 1970 00:00:00 UTC, ignoring leap seconds, in network byte order.

Signature: Contains the cryptographic signature that covers the entire record. The Record TTL and the sig fields are set to zero for the purpose of computing the Signature

## Appendix C. Encapsulated Control Message Format



"D" is the "DDT-originated" flag and is set by a DDT client to indicate that the receiver can and should return Map-Referral messages as appropriate.

Authors' Addresses

Vince Fuller  
cisco Systems  
Tasman Drive  
San Jose, CA 95134  
USA

Email: vaf@cisco.com

Darrel Lewis  
cisco Systems  
Tasman Drive  
San Jose, CA 95134  
USA

Email: darlewis@cisco.com

Vina Ermagan  
cisco Systems  
Tasman Drive  
San Jose, CA 95134  
USA

Email: vermagan@cisco.com

Amit Jain  
cisco Systems  
Tasman Drive  
San Jose, CA 95134  
USA

Email: amijain@cisco.com





Network Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: March 17, 2014

G. Schudel  
cisco Systems  
A. Jain  
Juniper Networks  
V. Moreno  
cisco Systems  
September 13, 2013

LISP MIB  
draft-ietf-lisp-mib-13

## Abstract

This document defines the MIB module that contains managed objects to support the monitoring devices that support the Locator/ID Separation Protocol (LISP). These objects provide information useful for monitoring LISP devices, including determining basic LISP configuration information, LISP functional status, and operational counters and other statistics.

## 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 <http://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 March 17, 2014.

## Copyright Notice

Copyright (c) 2013 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 (<http://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. Requirements Notation . . . . .	3
3. The Internet-Standard Management Framework . . . . .	3
4. Definition of Terms . . . . .	4
5. LISP MIB Objectives . . . . .	4
6. Structure of LISP MIB Module . . . . .	5
6.1. Overview of Defined Notifications . . . . .	5
6.2. Overview of Defined Tables . . . . .	5
7. LISP MIB Definitions . . . . .	6
8. Relationship to Other MIB Modules . . . . .	62
8.1. MIB modules required for IMPORTS . . . . .	62
9. Security Considerations . . . . .	62
10. IANA Considerations . . . . .	63
11. References . . . . .	63
11.1. Normative References . . . . .	63
11.2. Informative References . . . . .	64
Appendix A. Acknowledgments . . . . .	64

## 1. Introduction

This document describes the Management Information Base (MIB) module for use with network management protocols in the Internet community. Specifically, the MIB for managing devices that support the Locator/ID Separation Protocol (LISP) is described.

LISP [RFC6830] specifies a network-based architecture and mechanisms that implement a new semantic for IP addressing using two separate name spaces: Endpoint Identifiers (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.

From a data plane perspective, LISP traffic is handled exclusively at the network layer by devices performing Ingress Tunnel Router (ITR) and Egress Tunnel Router (ETR) LISP functions. Data plane operations performed by these devices are described in [RFC6830]. Additionally, data plane interworking between legacy (Internet) and LISP sites is implemented by devices performing Proxy ITR (PITR) and Proxy ETR (PETR) functions. The data plane operations of these devices is described in [RFC6832].

From a control plane perspective, LISP employs mechanisms related to creating, maintaining, and resolving mappings from EIDs to RLOCs. LISP ITRs, ETRs, PITRs, and PETRs perform specific control plane functions, and these control plane operations are described in [RFC6830]. Additionally, LISP infrastructure devices supporting LISP control plane functionality include Map-Servers and Map-Resolvers, and the control plane operations of these devices are described in [RFC6833].

## 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 [RFC2119].

## 3. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of RFC 3410 [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP).

Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIv2, which is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580 [RFC2580].

#### 4. Definition of Terms

This document does not define any new terms. All terms used in this document are listed here for completeness; the authoritative definition of each term can be found in the definition section of the respective, specified reference.

Endpoint ID (EID): [RFC6830]

Routing Locator (RLOC): [RFC6830]

EID-to-RLOC Cache: [RFC6830]

EID-to-RLOC Database: [RFC6830]

Ingress Tunnel Router (ITR): [RFC6830]

Egress Tunnel Router (ETR): [RFC6830]

xTR: [RFC6830]

Proxy ITR (PITR): [RFC6832]

Proxy ETR (PETR): [RFC6832]

LISP Site: [RFC6830]

Map-Server: [RFC6833]

Map-Resolver: [RFC6833]

Map-Request: [RFC6833]

Map-Reply: [RFC6833]

Negative Map-Reply: [RFC6833]

#### 5. LISP MIB Objectives

The objectives for this LISP MIB module are to provide a read-only mechanism to support the following functions:

- o Provide a means for obtaining (read-only) a current status of LISP features enabled on a device, and (read-only) a current status of configuration attributes related to those features. As one example, this MIB could determine the ON/OFF status of LISP features such as ITR, ETR, PITR, PETR, MS or MR support, specifically as related to both IPv4 or IPv6 address families. Other examples could include: obtaining the (read-only) status of whether rloc-probing is enabled, whether the use of a PETR is configured, and obtaining the (read-only) values of other related attributes such as the map-cache limit value, or a mapping time-to-live value.
- o Provide a means for obtaining (read-only) the current attributes of various LISP tables, such as the EID-to-RLLOC policy data contained in the Map-Cache, or the local EID-to-RLLOC policy data contained in the Mapping-Database.
- o Provide a means for obtaining (read-only) the current operational statistics of various LISP functions, such as the number of packets encapsulated and decapsulated by the device. Other counters of operational interest, depending on LISP function, include things like the current number of map-cache entries, and the total number and rate of map-requests received and sent by the device.

## 6. Structure of LISP MIB Module

### 6.1. Overview of Defined Notifications

No LISP MIB notifications are defined.

### 6.2. Overview of Defined Tables

The LISP MIB module is composed of the following tables of objects:

`lispFeatures` - This table provides information representing the various lisp features that can be enabled on LISP devices.

`lispIidToVrf` - This table provides information representing the mapping of a LISP instance ID to a VRF (Virtual Routing/Forwarding).

`lispGlobalStats` - This table provides global statistics for a given Instance ID per address-family on a LISP device.

`lispMappingDatabase` - This table represents the EID-to-RLOC database that contains the EID-prefix to RLOC mappings configured on an ETR. In general, this table would be representative of all such mappings for a given site that this device belongs to.

`lispMappingDatabaseLocator` - This table represents the set of routing locators contained in the EID-to-RLOC database configured on an ETR.

`lispMapCache` - This table represents the short-lived, on-demand table maintained on an ITR that stores, tracks, and times-out EID-to-RLOC mappings.

`lispMapCacheLocator` - This table represents the set of locators per EID prefix contained in the map-cache table of an ITR.

`lispConfiguredLocator` - This table represents the set of routing locators configured on a LISP device.

`lispEidRegistration` - This table provides the properties of each EID prefix that is registered with this device when configured to be a Map-Server.

`lispEidRegistrationEtr` - This table provides the properties of the different ETRs that send registers, for a given EID prefix, to this device when configured to be a Map-Server.

`lispEidRegistrationLocator` - This table provides the properties of the different locators per EID prefix that is registered with this device when configured to be a Map-Server.

`lispUseMapServer` - This table provides the properties of all Map-Servers that this device is configured to use.

`lispUseMapResolver` - This table provides the properties of all Map-Resolvers that this device is configured to use.

`lispUseProxyEtr` - This table provides the properties of all Proxy ETRs that this device is configured to use.

## 7. LISP MIB Definitions

```
LISP-MIB DEFINITIONS ::= BEGIN
```

```
IMPORTS
```

```
    MODULE-IDENTITY, OBJECT-TYPE,  
    mib-2, Unsigned32, Counter64,
```

```

Integer32, TimeTicks          FROM SNMPv2-SMI          -- [RFC2578]
TruthValue, TEXTUAL-CONVENTION,
TimeStamp                     FROM SNMPv2-TC           -- [RFC2579]
MODULE-COMPLIANCE, OBJECT-GROUP FROM SNMPv2-CONF      -- [RFC2580]
MplsL3VpnName
    FROM MPLS-L3VPN-STD-MIB          -- [RFC4382]
AddressFamilyNumbers
    FROM IANA-ADDRESS-FAMILY-NUMBERS-MIB;  --
    http://www.iana.org/assignments/ianaaddressfamilynumbers-mib

lispMIB MODULE-IDENTITY
    LAST-UPDATED "201309130000Z" -- 13 September 2013
    ORGANIZATION
        "IETF Locator/ID Separation Protocol (LISP) Working Group"
    CONTACT-INFO
        "Email: lisp@ietf.org
        WG charter:
        http://www.ietf.org/html.charters/lisp-charter.html"
    DESCRIPTION
        "This MIB module contains managed objects to support
        monitoring devices that support the Locator/ID Separation
        Protocol (LISP).

        Copyright (C) The IETF Trust (2013)."
    REVISION      "201309130000Z" -- 13 September 2013
    DESCRIPTION   "Initial version of the IETF LISP-MIB module. Published
        as RFC xxxx."
-- RFC Ed.: RFC-editor pls fill in xxxx
    ::= { mib-2 XXX }
-- RFC Ed.: assigned by IANA, see section 10 for details

--
-- Textual Conventions
--

LispAddressType ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "39a"
    STATUS current
    DESCRIPTION
        "LISP architecture can be applied to a wide variety of
        address-families. This textual-convention is a generalization
        for representing addresses belonging to those address-families.
        For convenience, this document refers to any such address as a
        LISP address. LispAddressType textual-convention consists of
        the following four-tuple:
        1. IANA Address Family Number: A field of length 2-octets,
           whose value is of the form following the assigned
           AddressFamilyNumbers textual-convention described in

```



- IANA-ADDRESS-FAMILY-NUMBERS-MIB DEFINITIONS [IANA]  
<http://www.iana.org/assignments/ianaaddressfamilynumbers-mib>.  
The enumerations are also listed in [IANA]. Note that this list of address family numbers is maintained by IANA.
2. Length of LISP address: A field of length 1-octet, whose value indicates the octet-length of the next (third) field of this LispAddressType four-tuple.
  3. LISP address: A field of variable length as indicated in the previous (second) field, whose value is an address of the IANA Address Family indicated in the first field of this LispAddressType four-tuple. Note that any of the IANA Address Families can be represented. Particularly when the address family is LISP Canonical Address Format (LCAF) [LCAF]  
<http://tools.ietf.org/id/draft-ietf-lisp-lcaf-02.txt> with IANA assigned Address Family Number 16387, then the first octet of this field indicates the LCAF type, and the rest of this field is same as the encoding format of the LISP Canonical Address after the length field, as defined in [LCAF].
  4. Mask-length of address: A field of length 1-octet, whose value is the mask-length to be applied to the LISP address specified in the previous (third) field.

To illustrate the use of this object, consider the LISP MIB Object below entitled lispMapCacheEntry. This object begins with the following entities:

```
lispMapCacheEntry ::= SEQUENCE {  
    lispMapCacheEidLength      INTEGER,  
    lispMapCacheEid            LispAddressType,  
    ... [skip] ...
```

Example 1: Suppose that the IPv4 EID prefix stored is 192.0.2.0/24. In this case, the values within lispMapCacheEntry would be:

```
lispMapCacheEidLength = 8  
lispMapCacheEid = 1, 4, 192.0.2.0, 24  
... [skip] ...
```

where 8 is the total length in octets of the next object (lispMapCacheEID of type LispAddressType). Then, the value 1 indicates the IPv4 AF (per [IANA]), the value 4 indicates that the AF is 4-octets in length, 192.0.2.0 is the IPv4 address, and the value 24 is the mask-length in bits. Note that the lispMapCacheEidLength value of 8 is used to compute the length of the fourth

(last) field in `lispMapCacheEid` to be 1 octet - as computed by  $8 - (2 + 1 + 4) = 1$ .

Example 2: Suppose that the IPv6 EID prefix stored is `2001:db8:a::/48`. In this case, the values within `lispMapCacheEntry` would be:

```
lispMapCacheEidLength = 20
lispMapCacheEid = 2, 16, 2001:db8:a::, 48
... [skip] ...
```

where 20 is the total length in octets of the next object (`lispMapCacheEID` of type `LispAddressType`). Then, the value 2 indicates the IPv4 AF (per [IANA]), the value 16 indicates that the AF is 16-octets in length, `2001:db8:a::` is the IPv6 address, and the value 48 is the mask-length in bits. Note that the `lispMapCacheEidLength` value of 20 is used to compute the length of the fourth (last) field in `lispMapCacheEid` to be 1 octet - as computed by  $20 - (2 + 1 + 16) = 1$ .

Example 3: As an example where LCAF is used, suppose that the IPv4 EID prefix stored is `192.0.2.0/24` and it is part of LISP Instance ID 101. In this case, the values within `lispMapCacheEntry` would be:

```
lispMapCacheEidLength = 11
lispMapCacheEid = 16387, 7, 2, 101, 1, 192.0.2.0, 24
... [skip] ...
```

where 11 is the total length in octets of the next object (`lispMapCacheEID` of type `LispAddressType`). Then, the value 16387 indicates the LCAF AF (see [IANA]), the value 7 indicates that the LCAF AF is 7-octets in length in this case, 2 indicates that LCAF Type 2 encoding is used (see [LCAF]), 101 gives the Instance ID, 1 gives the AFI (per [IANA]) for an IPv4 address, `192.0.2.0` is the IPv4 address, and 24 is the mask-length in bits. Note that the `lispMapCacheEidLength` value of 11 octets is used to compute the length of the last field in `lispMapCacheEid` to be 1 octet, as computed by  $11 - (2 + 1 + 1 + 1 + 1 + 4) = 1$ .

Note: all LISP header formats and locations of specific flags, bits, and fields are as given in the base LISP references of RFC6830, RFC6832, and RFC6833."

#### REFERENCE

"RFC6830, Section 14.2, draft-ietf-lisp-lcaf-02.txt."

SYNTAX OCTET STRING (SIZE (5..39))

--

-- Top level components of this MIB.

--

lispObjects OBJECT IDENTIFIER ::= { lispMIB 1 }

lispConformance OBJECT IDENTIFIER ::= { lispMIB 2 }

lispFeaturesTable OBJECT-TYPE

SYNTAX SEQUENCE OF LispFeaturesEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"This table represents the ON/OFF status of the various LISP features that can be enabled on LISP devices."

REFERENCE

"RFC6830, Section 4.0., Section 5.5., Section 6.3."

::= { lispObjects 1 }

lispFeaturesEntry OBJECT-TYPE

SYNTAX LispFeaturesEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"An entry (conceptual row) in the lispFeaturesTable."

INDEX { lispFeaturesInstanceID,  
lispFeaturesAddressFamily }

::= { lispFeaturesTable 1 }

LispFeaturesEntry ::= SEQUENCE {

lispFeaturesInstanceID	Unsigned32,
lispFeaturesAddressFamily	AddressFamilyNumbers,
lispFeaturesItrEnabled	TruthValue,
lispFeaturesEtrEnabled	TruthValue,
lispFeaturesProxyItrEnabled	TruthValue,
lispFeaturesProxyEtrEnabled	TruthValue,
lispFeaturesMapServerEnabled	TruthValue,
lispFeaturesMapResolverEnabled	TruthValue,
lispFeaturesMapCacheSize	Unsigned32,
lispFeaturesMapCacheLimit	Unsigned32,
lispFeaturesEtrMapCacheTtl	Unsigned32,
lispFeaturesRlocProbeEnabled	TruthValue,
lispFeaturesEtrAcceptMapDataEnabled	TruthValue,
lispFeaturesEtrAcceptMapDataVerifyEnabled	TruthValue,
lispFeaturesRouterTimeStamp	TimeStamp

```
}

lispFeaturesInstanceID OBJECT-TYPE
    SYNTAX      Unsigned32 (0..16777215)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This represents the Instance ID of the LISP header.
        An Instance ID in the LISP address encoding helps
        uniquely identify the AFI-based address space to which
        a given EID belongs. It's default value is 0."
    DEFVAL { 0 }
    ::= { lispFeaturesEntry 1 }

lispFeaturesAddressFamily OBJECT-TYPE
    SYNTAX      AddressFamilyNumbers
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The IANA address family number of destination address
        of packets that this LISP device is enabled to process."
    ::= { lispFeaturesEntry 2 }

lispFeaturesItrEnabled OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates the status of ITR role on this device. If
        this object is true, then ITR feature is enabled."
    ::= { lispFeaturesEntry 3 }

lispFeaturesEtrEnabled OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates the status of ETR role on this device. If
        this object is true, then ETR feature is enabled."
    ::= { lispFeaturesEntry 4 }

lispFeaturesProxyItrEnabled OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates the status of Proxy-ITR role on this device.
        If this object is true, then Proxy-ITR feature is enabled."
```

```
 ::= { lispFeaturesEntry 5 }

lispFeaturesProxyEtrEnabled OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates the status of Proxy-ETR role on this device.
        If this object is true, then Proxy-ETR feature is enabled."
    ::= { lispFeaturesEntry 6 }

lispFeaturesMapServerEnabled OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates the status of Map Server role on this device.
        If this object is true, then Map Server feature is
        enabled."
    ::= { lispFeaturesEntry 7 }

lispFeaturesMapResolverEnabled OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates the status of Map Resolver role on this device.
        If this object is true, then Map Resolver feature is
        enabled."
    ::= { lispFeaturesEntry 8 }

lispFeaturesMapCacheSize OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Size of EID-to-RLOC map cache on this device."
    ::= { lispFeaturesEntry 9 }

lispFeaturesMapCacheLimit OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Maximum permissible entries in EID-to-RLOC map cache on
        this device."
    ::= { lispFeaturesEntry 10 }
```

```
lispFeaturesEtrMapCacheTtl OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The stored Record TTL of the EID-to-RLOC map record in
        the map cache."
    ::= { lispFeaturesEntry 11 }

lispFeaturesRlocProbeEnabled OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates the status of rloc-probing feature on this
        device.  If this object is true, then this feature is
        enabled."
    ::= { lispFeaturesEntry 12 }

lispFeaturesEtrAcceptMapDataEnabled OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates the status of accepting piggybacked mapping
        data received in a map-request on this device.  If this
        object is true, then this device accepts piggybacked
        mapping data."
    ::= { lispFeaturesEntry 13 }

lispFeaturesEtrAcceptMapDataVerifyEnabled OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates the status of verifying accepted piggybacked
        mapping data received in a map-request on this device.
        If this object is true, then this device verifies
        accepted piggybacked mapping data."
    ::= { lispFeaturesEntry 14 }

lispFeaturesRouterTimeStamp OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The value of sysUpTime at which LISP feature was
        enabled on this device."
```

If this information was present at the most recent re-initialization of the local management subsystem, then this object contains a zero value."  
DEFVAL { 0 }  
::= { lispFeaturesEntry 15 }

lispIidToVrfTable OBJECT-TYPE  
SYNTAX SEQUENCE OF LispIidToVrfEntry  
MAX-ACCESS not-accessible  
STATUS current  
DESCRIPTION  
"This table represents the mapping of LISP Instance ID to a VRF."  
REFERENCE  
"RFC6830, Section 5.5. and RFC4382, Section 7."  
::= { lispObjects 2 }

lispIidToVrfEntry OBJECT-TYPE  
SYNTAX LispIidToVrfEntry  
MAX-ACCESS not-accessible  
STATUS current  
DESCRIPTION  
"An entry (conceptual row) in the lispIidToVrfTable."  
INDEX { lispFeaturesInstanceID }  
::= { lispIidToVrfTable 1 }

LispIidToVrfEntry ::= SEQUENCE {  
lispIidToVrfName MplsL3VpnName  
}

lispIidToVrfName OBJECT-TYPE  
SYNTAX MplsL3VpnName  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
"The identifier for each VPN that is mapped to the given LISP Instance ID."  
::= { lispIidToVrfEntry 1 }

lispGlobalStatsTable OBJECT-TYPE  
SYNTAX SEQUENCE OF LispGlobalStatsEntry  
MAX-ACCESS not-accessible  
STATUS current  
DESCRIPTION  
"This table provides global statistics for a given

Instance ID per address-family on a LISP device."

REFERENCE  
"RFC6830, Section 6.1."  
::= { lispObjects 3 }

lispGlobalStatsEntry OBJECT-TYPE  
SYNTAX LispGlobalStatsEntry  
MAX-ACCESS not-accessible  
STATUS current  
DESCRIPTION  
"An entry (conceptual row) in the  
lispGlobalStatsTable."  
INDEX { lispFeaturesInstanceID,  
lispFeaturesAddressFamily }  
::= { lispGlobalStatsTable 1 }

LispGlobalStatsEntry ::= SEQUENCE {  
lispGlobalStatsMapRequestsIn Counter64,  
lispGlobalStatsMapRequestsOut Counter64,  
lispGlobalStatsMapRepliesIn Counter64,  
lispGlobalStatsMapRepliesOut Counter64,  
lispGlobalStatsMapRegistersIn Counter64,  
lispGlobalStatsMapRegistersOut Counter64  
}

lispGlobalStatsMapRequestsIn OBJECT-TYPE  
SYNTAX Counter64  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
"Total number of map requests received by this device for  
any EID prefix of the given address family and Instance ID.  
  
Discontinuities in this monotonically increasing value occur  
at re-initialization of the management system.  
Discontinuities can also occur as a result of LISP features  
being removed, which can be detected by observing the value  
of lispFeaturesRouterTimeStamp."  
::= { lispGlobalStatsEntry 1 }

lispGlobalStatsMapRequestsOut OBJECT-TYPE  
SYNTAX Counter64  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
"Total number of map requests sent by this device for any  
EID prefix of the given address family and Instance ID."



Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of LISP features being removed, which can be detected by observing the value of lispFeaturesRouterTimeStamp."

::= { lispGlobalStatsEntry 2 }

lispGlobalStatsMapRepliesIn OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"Total number of map replies received by this device for any EID prefix of the given address family and Instance ID.

Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of LISP features being removed, which can be detected by observing the value of lispFeaturesRouterTimeStamp."

::= { lispGlobalStatsEntry 3 }

lispGlobalStatsMapRepliesOut OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"Total number of map replies sent by this device for any EID prefix of the given address family and Instance ID.

Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of LISP features being removed, which can be detected by observing the value of lispFeaturesRouterTimeStamp."

::= { lispGlobalStatsEntry 4 }

lispGlobalStatsMapRegistersIn OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"Total number of map registers received by this device for any EID prefix of the given address family and Instance ID.

Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of LISP features

being removed, which can be detected by observing the value of lispFeaturesRouterTimeStamp."

::= { lispGlobalStatsEntry 5 }

lispGlobalStatsMapRegistersOut OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"Total number of map registers sent by this device for any EID prefix of the given address family and Instance ID.

Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of LISP features being removed, which can be detected by observing the value of lispFeaturesRouterTimeStamp."

::= { lispGlobalStatsEntry 6 }

lispMappingDatabaseTable OBJECT-TYPE

SYNTAX SEQUENCE OF LispMappingDatabaseEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"This table represents the EID-to-RLOC mapping database that contains the EID-prefix to RLOC mappings configured on an ETR.

This table represents all such mappings for the given LISP site to which this device belongs."

REFERENCE

"RFC6830, Section 6.0."

::= { lispObjects 4 }

lispMappingDatabaseEntry OBJECT-TYPE

SYNTAX LispMappingDatabaseEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"An entry (conceptual row) in lispMappingDatabaseTable."

INDEX { lispMappingDatabaseEidLength,  
lispMappingDatabaseEid }

::= { lispMappingDatabaseTable 1 }

LispMappingDatabaseEntry ::= SEQUENCE {

lispMappingDatabaseEidLength Integer32,

```
    lispMappingDatabaseEid          LispAddressType,
    lispMappingDatabaseLsb          Unsigned32,
    lispMappingDatabaseEidPartitioned TruthValue,
    lispMappingDatabaseTimeStamp    TimeStamp,
    lispMappingDatabaseDecapOctets   Counter64,
    lispMappingDatabaseDecapPackets  Counter64,
    lispMappingDatabaseEncapOctets   Counter64,
    lispMappingDatabaseEncapPackets  Counter64
}

lispMappingDatabaseEidLength OBJECT-TYPE
    SYNTAX      Integer32 (5..39)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object gives the octet-length of
         lispMappingDatabaseEid."
    ::= { lispMappingDatabaseEntry 1 }

lispMappingDatabaseEid OBJECT-TYPE
    SYNTAX      LispAddressType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The EID prefix of the mapping database."
    ::= { lispMappingDatabaseEntry 2 }

lispMappingDatabaseLsb OBJECT-TYPE
    SYNTAX      Unsigned32 (0..4294967295)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The locator status bits for this EID prefix."
    ::= { lispMappingDatabaseEntry 3 }

lispMappingDatabaseEidPartitioned OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only

    STATUS      current
    DESCRIPTION
        "Indicates if this device is partitioned from the site that
         contains this EID prefix. If this object is true, then it
         means this device is partitioned from the site."
    ::= { lispMappingDatabaseEntry 4 }

lispMappingDatabaseTimeStamp OBJECT-TYPE
    SYNTAX      TimeStamp
```

```
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
    "The value of sysUpTime at which the EID Prefix information
    represented by this mapping database entry was configured
    on this device.

    If this information was present at the most recent
    re-initialization of the local management subsystem, then
    this object contains a zero value."
DEFVAL { 0 }
 ::= { lispMappingDatabaseEntry 5 }

lispMappingDatabaseDecapOctets OBJECT-TYPE
SYNTAX      Counter64
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
    "The number of octets, after decapsulation, of LISP packets
    that were decapsulated by this device addressed to a host
    within this EID-prefix.

    Discontinuities in this monotonically increasing value occur
    at re-initialization of the management system.
    Discontinuities can also occur as a result of LISP features
    being removed, which can be detected by observing the value
    of lispMappingDatabaseTimeStamp."
 ::= { lispMappingDatabaseEntry 6 }

lispMappingDatabaseDecapPackets OBJECT-TYPE
SYNTAX      Counter64
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
    "The number of LISP packets that were decapsulated by this
    device addressed to a host within this EID-prefix.

    Discontinuities in this monotonically increasing value occur
    at re-initialization of the management system.
    Discontinuities can also occur as a result of LISP features
    being removed, which can be detected by observing the value
    of lispMappingDatabaseTimeStamp."
 ::= { lispMappingDatabaseEntry 7 }

lispMappingDatabaseEncapOctets OBJECT-TYPE
SYNTAX      Counter64
MAX-ACCESS read-only
STATUS      current
```

## DESCRIPTION

"The number of octets, before encapsulation, of LISP packets that were encapsulated by this device, whose inner header source address matched this EID prefix.

Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of LISP features being removed, which can be detected by observing the value of lispMappingDatabaseTimeStamp."

::= { lispMappingDatabaseEntry 8 }

## lispMappingDatabaseEncapPackets OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

## DESCRIPTION

"The number of LISP packets that were encapsulated by this device whose inner header source address matched this EID prefix.

Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of LISP features being removed, which can be detected by observing the value of lispMappingDatabaseTimeStamp."

::= { lispMappingDatabaseEntry 9 }

## lispMappingDatabaseLocatorTable OBJECT-TYPE

SYNTAX SEQUENCE OF LispMappingDatabaseLocatorEntry

MAX-ACCESS not-accessible

STATUS current

## DESCRIPTION

"This table represents the set of routing locators per EID prefix contained in the EID-to-RLOC database configured on this ETR."

## REFERENCE

"RFC6830, Section 6.2."

::= { lispObjects 5 }

## lispMappingDatabaseLocatorEntry OBJECT-TYPE

SYNTAX LispMappingDatabaseLocatorEntry

MAX-ACCESS not-accessible

STATUS current

## DESCRIPTION

"An entry (conceptual row) in the lispMappingDatabaseLocatorTable."

```
INDEX { lispMappingDatabaseEidLength,
        lispMappingDatabaseEid,
        lispMappingDatabaseLocatorRlocLength,
        lispMappingDatabaseLocatorRloc }
 ::= { lispMappingDatabaseLocatorTable 1 }

LispMappingDatabaseLocatorEntry ::= SEQUENCE {
    lispMappingDatabaseLocatorRlocLength      Integer32,
    lispMappingDatabaseLocatorRloc           LispAddressType,
    lispMappingDatabaseLocatorRlocPriority    Integer32,
    lispMappingDatabaseLocatorRlocWeight     Integer32,
    lispMappingDatabaseLocatorRlocMPriority  Integer32,
    lispMappingDatabaseLocatorRlocMWeight   Integer32,
    lispMappingDatabaseLocatorRlocState      INTEGER,
    lispMappingDatabaseLocatorRlocLocal      INTEGER,
    lispMappingDatabaseLocatorRlocTimeStamp  TimeStamp,
    lispMappingDatabaseLocatorRlocDecapOctets Counter64,
    lispMappingDatabaseLocatorRlocDecapPackets Counter64,
    lispMappingDatabaseLocatorRlocEncapOctets Counter64,
    lispMappingDatabaseLocatorRlocEncapPackets Counter64
}

lispMappingDatabaseLocatorRlocLength OBJECT-TYPE
    SYNTAX      Integer32 (5..39)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object is used to get the octet-length of
        lispMappingDatabaseLocatorRloc."
    ::= { lispMappingDatabaseLocatorEntry 1 }

lispMappingDatabaseLocatorRloc OBJECT-TYPE
    SYNTAX      LispAddressType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object is a locator for the given EID prefix in
        the mapping database."
    ::= { lispMappingDatabaseLocatorEntry 2 }

lispMappingDatabaseLocatorRlocPriority OBJECT-TYPE
    SYNTAX      Integer32 (0..255)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The unicast priority of the RLOC."
    ::= { lispMappingDatabaseLocatorEntry 3 }
```

```
lispMappingDatabaseLocatorRlocWeight OBJECT-TYPE
    SYNTAX      Integer32 (0..100)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The unicast weight of the RLOC."
    ::= { lispMappingDatabaseLocatorEntry 4 }

lispMappingDatabaseLocatorRlocMPriority OBJECT-TYPE
    SYNTAX      Integer32 (0..255)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The multicast priority of the RLOC."
    ::= { lispMappingDatabaseLocatorEntry 5 }

lispMappingDatabaseLocatorRlocMWeight OBJECT-TYPE
    SYNTAX      Integer32 (0..100)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The multicast weight of the RLOC."
    ::= { lispMappingDatabaseLocatorEntry 6 }

lispMappingDatabaseLocatorRlocState OBJECT-TYPE
    SYNTAX      INTEGER {
                    up (1),
                    down (2),
                    unreachable (3)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The state of this RLOC as per this device.
        (1 = RLOC is up; 2 = RLOC is down; 3 = RLOC is unreachable)."
    ::= { lispMappingDatabaseLocatorEntry 7 }

lispMappingDatabaseLocatorRlocLocal OBJECT-TYPE
    SYNTAX      INTEGER {
                    siteself (1),
                    sitelocal (2)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates whether the RLOC is local to this device
        (or remote, meaning local to another device in the same LISP
        site). (1 = RLOC is an address on this device; 2 = RLOC is
```

```
        an address on another device)."  
 ::= { lispMappingDatabaseLocatorEntry 8 }  
  
lispMappingDatabaseLocatorRlocTimeStamp OBJECT-TYPE  
    SYNTAX      TimeStamp  
    MAX-ACCESS  read-only  
    STATUS      current  
    DESCRIPTION  
        "The value of sysUpTime at which the RLOC of the EID Prefix  
        represented by this mapping database entry was configured  
        on this device.  
  
        If this information was present at the most recent  
        re-initialization of the local management subsystem, then  
        this object contains a zero value."  
    DEFVAL { 0 }  
 ::= { lispMappingDatabaseLocatorEntry 9 }  
  
lispMappingDatabaseLocatorRlocDecapOctets OBJECT-TYPE  
    SYNTAX      Counter64  
    MAX-ACCESS  read-only  
    STATUS      current  
    DESCRIPTION  
        "The number of octets of LISP packets that were  
        addressed to this RLOC of the EID-prefix and  
        were decapsulated.  
  
        Discontinuities in this monotonically increasing value occur  
        at re-initialization of the management system.  
        Discontinuities can also occur as a result of database  
        mappings getting re-configured or RLOC status changes, which  
        can be detected by observing the value of  
        lispMappingDatabaseLocatorRlocTimeStamp."  
 ::= { lispMappingDatabaseLocatorEntry 10 }  
  
lispMappingDatabaseLocatorRlocDecapPackets OBJECT-TYPE  
    SYNTAX      Counter64  
    MAX-ACCESS  read-only  
    STATUS      current  
    DESCRIPTION  
        "The number of LISP packets that were addressed to this RLOC  
        of the EID-prefix and were decapsulated.  
  
        Discontinuities in this monotonically increasing value occur  
        at re-initialization of the management system.  
        Discontinuities can also occur as a result of database  
        mappings getting re-configured or RLOC status changes, which  
        can be detected by observing the value of
```



```
    lispMappingDatabaseLocatorRlocTimeStamp."
 ::= { lispMappingDatabaseLocatorEntry 11 }
```

**lispMappingDatabaseLocatorRlocEncapOctets OBJECT-TYPE**  
SYNTAX Counter64  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
    "The number of octets of LISP packets that were encapsulated  
    by this device using this RLOC address as the source, and  
    that were sourced by an address of this EID-prefix.

Discontinuities in this monotonically increasing value occur  
at re-initialization of the management system.  
Discontinuities can also occur as a result of database  
mappings getting re-configured or RLOC status changes, which  
can be detected by observing the value of  
lispMappingDatabaseLocatorRlocTimeStamp."  
 ::= { lispMappingDatabaseLocatorEntry 12 }

**lispMappingDatabaseLocatorRlocEncapPackets OBJECT-TYPE**  
SYNTAX Counter64  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
    "The number of LISP packets that were encapsulated by this  
    device using this RLOC address as the source, and that were  
    sourced by an address of this EID-prefix.

Discontinuities in this monotonically increasing value occur  
at re-initialization of the management system.  
Discontinuities can also occur as a result of database  
mappings getting re-configured or RLOC status changes, which  
can be detected by observing the value of  
lispMappingDatabaseLocatorRlocTimeStamp."  
 ::= { lispMappingDatabaseLocatorEntry 13 }

**lispMapCacheTable OBJECT-TYPE**  
SYNTAX SEQUENCE OF LispMapCacheEntry  
MAX-ACCESS not-accessible  
STATUS current  
DESCRIPTION  
    "This table represents the short-lived, on-demand table on  
    an ITR that stores, tracks, and is responsible for  
    timing-out and otherwise validating EID-to-RLOC mappings."  
REFERENCE  
    "RFC6830, Section 6.0., Section 12.0."

```
 ::= { lispObjects 6 }

lispMapCacheEntry OBJECT-TYPE
    SYNTAX      LispMapCacheEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry (conceptual row) in the
        lispMapCacheTable."
    INDEX       { lispMapCacheEidLength,
                  lispMapCacheEid }
    ::= { lispMapCacheTable 1 }

LispMapCacheEntry ::= SEQUENCE {
    lispMapCacheEidLength      Integer32,
    lispMapCacheEid            LispAddressType,
    lispMapCacheEidTimeStamp   TimeStamp,
    lispMapCacheEidExpiryTime  TimeTicks,
    lispMapCacheEidState       TruthValue,
    lispMapCacheEidAuthoritative TruthValue,
    lispMapCacheEidDecapOctets Counter64,
    lispMapCacheEidDecapPackets Counter64,
    lispMapCacheEidEncapOctets Counter64,
    lispMapCacheEidEncapPackets Counter64
}

lispMapCacheEidLength OBJECT-TYPE
    SYNTAX      Integer32 (5..39)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object is used to get the octet-length of
        lispMapCacheEid."
    ::= { lispMapCacheEntry 1 }

lispMapCacheEid OBJECT-TYPE
    SYNTAX      LispAddressType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The EID prefix in the mapping cache."
    ::= { lispMapCacheEntry 2 }

lispMapCacheEidTimeStamp OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
```

"The value of sysUpTime at which the EID Prefix information represented by this entry was learned by this device.

If this information was present at the most recent re-initialization of the local management subsystem, then this object contains a zero value."

DEFVAL { 0 }

::= { lispMapCacheEntry 3 }

lispMapCacheEidExpiryTime OBJECT-TYPE

SYNTAX TimeTicks

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The time remaining before the ITR times-out this EID prefix."

::= { lispMapCacheEntry 4 }

lispMapCacheEidState OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object is used to indicate the activity of this EID prefix. If this object is true, then it means this EID prefix is seeing activity."

::= { lispMapCacheEntry 5 }

lispMapCacheEidAuthoritative OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object is used to indicate whether the EID prefix was installed by an authoritative map-reply. If this object is true, then it means this EID prefix was installed by an authoritative map-reply."

::= { lispMapCacheEntry 6 }

lispMapCacheEidDecapOctets OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of octets of LISP packets that were decapsulated by this device and were sourced from a remote host within this EID-prefix."

Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of cache being removed and replaced, which can be detected by observing the value of lispMapCacheEidTimeStamp."

::= { lispMapCacheEntry 7 }

lispMapCacheEidDecapPackets OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of LISP packets that were decapsulated by this device and were sourced from a remote host within this EID-prefix.

Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of cache being removed and replaced, which can be detected by observing the value of lispMapCacheEidTimeStamp."

::= { lispMapCacheEntry 8 }

lispMapCacheEidEncapOctets OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of octets of LISP packets that were encapsulated by this device using the given EID-prefix in the map cache.

Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of cache being removed and replaced, which can be detected by observing the value of lispMapCacheEidTimeStamp."

::= { lispMapCacheEntry 9 }

lispMapCacheEidEncapPackets OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of LISP packets that were encapsulated by this device using the given EID-prefix in the map cache.

Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of cache being removed and replaced, which can be detected by observing the value of `lispMapCacheEidTimeStamp`."

```
 ::= { lispMapCacheEntry 10 }
```

```
lispMapCacheLocatorTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LispMapCacheLocatorEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table represents the set of locators per EID prefix
        contained in the map-cache table of an ITR."
    REFERENCE
        "RFC6830, Section 6.3."
    ::= { lispObjects 7 }
```

```
lispMapCacheLocatorEntry OBJECT-TYPE
    SYNTAX      LispMapCacheLocatorEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry (conceptual row) in the
        lispMapCacheLocatorTable."
    INDEX       { lispMapCacheEidLength,
                  lispMapCacheEid,
                  lispMapCacheLocatorRlocLength,
                  lispMapCacheLocatorRloc }
    ::= { lispMapCacheLocatorTable 1 }
```

```
LispMapCacheLocatorEntry ::= SEQUENCE {
    lispMapCacheLocatorRlocLength      Integer32,
    lispMapCacheLocatorRloc            LispAddressType,
    lispMapCacheLocatorRlocPriority     Integer32,
    lispMapCacheLocatorRlocWeight      Integer32,
    lispMapCacheLocatorRlocMPriority   Integer32,
    lispMapCacheLocatorRlocMWeight     Integer32,
    lispMapCacheLocatorRlocState        INTEGER,
    lispMapCacheLocatorRlocTimeStamp    TimeStamp,
    lispMapCacheLocatorRlocLastPriorityChange TimeTicks,
    lispMapCacheLocatorRlocLastWeightChange TimeTicks,
    lispMapCacheLocatorRlocLastMPriorityChange TimeTicks,
    lispMapCacheLocatorRlocLastMWeightChange TimeTicks,
    lispMapCacheLocatorRlocLastStateChange TimeTicks,
    lispMapCacheLocatorRlocRtt          TimeTicks,
    lispMapCacheLocatorRlocDecapOctets   Counter64,
    lispMapCacheLocatorRlocDecapPackets Counter64,
    lispMapCacheLocatorRlocEncapOctets   Counter64,
```

```
        lispMapCacheLocatorRlocEncapPackets      Counter64
    }

    lispMapCacheLocatorRlocLength OBJECT-TYPE
        SYNTAX      Integer32 (5..39)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "This object is used to get the octet-length of
             lispMapCacheLocatorRloc."
        ::= { lispMapCacheLocatorEntry 1 }

    lispMapCacheLocatorRloc OBJECT-TYPE
        SYNTAX      LispAddressType
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "The locator for the EID prefix in the mapping cache."
        ::= { lispMapCacheLocatorEntry 2 }

    lispMapCacheLocatorRlocPriority OBJECT-TYPE
        SYNTAX      Integer32 (0..255)
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The unicast priority of the RLOC for this EID prefix
             (0-255); lower more preferred. "
        ::= { lispMapCacheLocatorEntry 3 }

    lispMapCacheLocatorRlocWeight OBJECT-TYPE
        SYNTAX      Integer32 (0..100)
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The unicast weight of the RLOC for this EID prefix
             (0 - 100) percentage. "
        ::= { lispMapCacheLocatorEntry 4 }

    lispMapCacheLocatorRlocMPriority OBJECT-TYPE
        SYNTAX      Integer32 (0..255)
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The multicast priority of the RLOC for this EID prefix
             (0-255); lower more preferred."
        ::= { lispMapCacheLocatorEntry 5 }

    lispMapCacheLocatorRlocMWeight OBJECT-TYPE
```

```
SYNTAX      Integer32 (0..100)
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The multicast weight of the RLOC for this EID prefix
    (0 - 100) percentage."
 ::= { lispMapCacheLocatorEntry 6 }

lispMapCacheLocatorRlocState OBJECT-TYPE
SYNTAX      INTEGER {
                up (1),
                down (2),
                unreachable (3)
            }
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The state of this RLOC as per this device
    (1 = RLOC is up; 2 = RLOC is down; 3 = RLOC is unreachable)."
```

```
 ::= { lispMapCacheLocatorEntry 7 }

lispMapCacheLocatorRlocTimeStamp OBJECT-TYPE
SYNTAX      TimeStamp
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The value of sysUpTime at which the RLOC of EID prefix
    information represented by this entry was learned by
    this device.

    If this information was present at the most recent
    re-initialization of the local management subsystem,
    then this object contains a zero value."
DEFVAL { 0 }
 ::= { lispMapCacheLocatorEntry 8 }

lispMapCacheLocatorRlocLastPriorityChange OBJECT-TYPE
SYNTAX      TimeTicks
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Time elapsed since the last change of the unicast priority
    of the RLOC for this EID prefix. Note that this is
    independent of lispMapCacheLocatorRlocTimeStamp."
 ::= { lispMapCacheLocatorEntry 9 }

lispMapCacheLocatorRlocLastWeightChange OBJECT-TYPE
SYNTAX      TimeTicks
```

```
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
    "Time elapsed since the last change of the unicast weight
    of the RLOC for this EID prefix. Note that this is
    independent of lispMapCacheLocatorRlocTimeStamp."
 ::= { lispMapCacheLocatorEntry 10 }

lispMapCacheLocatorRlocLastMPriorityChange OBJECT-TYPE
SYNTAX      TimeTicks
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
    "Time since the last change of the multicast priority of the
    RLOC for this EID prefix."
 ::= { lispMapCacheLocatorEntry 11 }

lispMapCacheLocatorRlocLastMWeightChange OBJECT-TYPE
SYNTAX      TimeTicks
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
    "Time since the last change of the multicast weight of the
    RLOC for this EID prefix."
 ::= { lispMapCacheLocatorEntry 12 }

lispMapCacheLocatorRlocLastStateChange OBJECT-TYPE
SYNTAX      TimeTicks
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
    "Time since the last change of the up/down state of the
    RLOC for this EID prefix."
 ::= { lispMapCacheLocatorEntry 13 }

lispMapCacheLocatorRlocRtt OBJECT-TYPE
SYNTAX      TimeTicks
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
    "Round trip time of RLOC probe and map-reply for this RLOC
    address for this prefix."
 ::= { lispMapCacheLocatorEntry 14 }

lispMapCacheLocatorRlocDecapOctets OBJECT-TYPE
SYNTAX      Counter64
MAX-ACCESS read-only
STATUS      current
```



## DESCRIPTION

"The number of octets of LISP packets that were decapsulated by this device and were sourced from a remote host within this EID-prefix and were encapsulated for this RLOC.

Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of RLOC of cache being removed and replaced, which can be detected by observing the value of lispMapCacheLocatorRlocTimeStamp."

::= { lispMapCacheLocatorEntry 15 }

## lispMapCacheLocatorRlocDecapPackets OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

## DESCRIPTION

"The number of LISP packets that were decapsulated by this device and were sourced from a remote host within this EID-prefix and were encapsulated for this RLOC.

Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of RLOC of cache being removed and replaced, which can be detected by observing the value of lispMapCacheLocatorRlocTimeStamp."

::= { lispMapCacheLocatorEntry 16 }

## lispMapCacheLocatorRlocEncapOctets OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

## DESCRIPTION

"The number of octets of LISP packets that matched this EID prefix and were encapsulated using this RLOC address.

Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of RLOC of cache being removed and replaced, which can be detected by observing the value of lispMapCacheLocatorRlocTimeStamp."

::= { lispMapCacheLocatorEntry 17 }

## lispMapCacheLocatorRlocEncapPackets OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

## DESCRIPTION

"The number of LISP packets that matched this EID prefix and were encapsulated using this RLOC address.

Discontinuities in this monotonically increasing value occur at re-initialization of the management system. Discontinuities can also occur as a result of RLOC of cache being removed and replaced, which can be detected by observing the value of lispMapCacheLocatorRlocTimeStamp."

```
::= { lispMapCacheLocatorEntry 18 }
```

lispConfiguredLocatorTable OBJECT-TYPE

SYNTAX SEQUENCE OF LispConfiguredLocatorEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"This table represents the set of routing locators configured on this device. Note that the Proxy-ITR configured addresses are treated as routing locators and therefore can be part of this table."

REFERENCE

"RFC6830, Section 6.3."

```
::= { lispObjects 8 }
```

lispConfiguredLocatorEntry OBJECT-TYPE

SYNTAX LispConfiguredLocatorEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"An entry (conceptual row) in the lispConfiguredLocatorTable."

INDEX { lispConfiguredLocatorRlocLength,  
lispConfiguredLocatorRloc }

```
::= { lispConfiguredLocatorTable 1 }
```

LispConfiguredLocatorEntry ::= SEQUENCE {

lispConfiguredLocatorRlocLength	Integer32,
lispConfiguredLocatorRloc	LispAddressType,
lispConfiguredLocatorRlocState	INTEGER,
lispConfiguredLocatorRlocLocal	INTEGER,
lispConfiguredLocatorRlocTimeStamp	TimeStamp,
lispConfiguredLocatorRlocDecapOctets	Counter64,
lispConfiguredLocatorRlocDecapPackets	Counter64,
lispConfiguredLocatorRlocEncapOctets	Counter64,
lispConfiguredLocatorRlocEncapPackets	Counter64

}

lispConfiguredLocatorRlocLength OBJECT-TYPE

```
SYNTAX      Integer32 (5..39)
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "This object is used to get the octet-length of
    lispConfiguredLocatorRloc."
 ::= { lispConfiguredLocatorEntry 1 }

lispConfiguredLocatorRloc OBJECT-TYPE
    SYNTAX      LispAddressType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object is a RLOC address configured on this device.
        It can be an RLOC that is local to this device or can be an
        RLOC which belongs to another ETR within the same site.
        Proxy-ITR address is treated as an RLOC."
    ::= { lispConfiguredLocatorEntry 2 }

lispConfiguredLocatorRlocState OBJECT-TYPE
    SYNTAX      INTEGER {
        up (1),
        down (2),
        unreachable (3)
    }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The state of this RLOC as per this device. (1 = RLOC is up;
        2 = RLOC is down; 3 = RLOC is unreachable)."
    ::= { lispConfiguredLocatorEntry 3 }

lispConfiguredLocatorRlocLocal OBJECT-TYPE
    SYNTAX      INTEGER {
        siteself (1),
        sitelocal (2)
    }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates whether the RLOC is local to this device (or
        remote, meaning local to another device in the same LISP
        site). (1 = RLOC is an address on this device; 2 = RLOC is
        an address on another device)."
    ::= { lispConfiguredLocatorEntry 4 }

lispConfiguredLocatorRlocTimeStamp OBJECT-TYPE
    SYNTAX      TimeStamp
```

```
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
    "The value of sysUpTime at which the RLOC was configured on
    this device.

    If this information was present at the most recent
    re-initialization of the local management subsystem, then
    this object contains a zero value."
DEFVAL { 0 }
 ::= { lispConfiguredLocatorEntry 5 }

lispConfiguredLocatorRlocDecapOctets OBJECT-TYPE
SYNTAX      Counter64
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
    "The number of octets of LISP packets that were addressed to
    this RLOC and were decapsulated.

    Discontinuities in this monotonically increasing value occur
    at re-initialization of the management system.
    Discontinuities can also occur as a result of configured
    RLOC being removed and replaced, which can be detected by
    observing the value of lispConfiguredLocatorRlocTimeStamp."
 ::= { lispConfiguredLocatorEntry 6 }

lispConfiguredLocatorRlocDecapPackets OBJECT-TYPE
SYNTAX      Counter64
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
    "The number of LISP packets that were addressed to this RLOC
    and were decapsulated.

    Discontinuities in this monotonically increasing value occur
    at re-initialization of the management system.
    Discontinuities can also occur as a result of configured
    RLOC being removed and replaced, which can be detected by
    observing the value of lispConfiguredLocatorRlocTimeStamp."
 ::= { lispConfiguredLocatorEntry 7 }

lispConfiguredLocatorRlocEncapOctets OBJECT-TYPE
SYNTAX      Counter64
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
    "The number of octets of LISP packets that were encapsulated
```

by this device using this RLOC address as the source.

Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of configured RLOC being removed and replaced, which can be detected by observing the value of lispConfiguredLocatorRlocTimeStamp."

::= { lispConfiguredLocatorEntry 8 }

lispConfiguredLocatorRlocEncapPackets OBJECT-TYPE

SYNTAX Counter64

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The number of LISP packets that were encapsulated by this device using this RLOC address as the source.

Discontinuities in this monotonically increasing value occur at re-initialization of the management system.

Discontinuities can also occur as a result of configured RLOC being removed and replaced, which can be detected by observing the value of lispConfiguredLocatorRlocTimeStamp."

::= { lispConfiguredLocatorEntry 9 }

lispEidRegistrationTable OBJECT-TYPE

SYNTAX SEQUENCE OF LispEidRegistrationEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"This table provides the properties of each LISP EID prefix that is registered with this device when configured to be a Map-Server."

REFERENCE

"RFC6833, Section 4.0."

::= { lispObjects 9 }

lispEidRegistrationEntry OBJECT-TYPE

SYNTAX LispEidRegistrationEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"An entry (conceptual row) in the lispEidRegistrationTable."

INDEX { lispEidRegistrationEidLength,  
lispEidRegistrationEid }

::= { lispEidRegistrationTable 1 }

```
LispEidRegistrationEntry ::= SEQUENCE {
    lispEidRegistrationEidLength      Integer32,
    lispEidRegistrationEid           LispAddressType,
    lispEidRegistrationSiteName      OCTET STRING,
    lispEidRegistrationSiteDescription OCTET STRING,
    lispEidRegistrationIsRegistered  TruthValue,
    lispEidRegistrationFirstTimeStamp TimeStamp,
    lispEidRegistrationLastTimeStamp TimeStamp,
    lispEidRegistrationLastRegisterSenderLength Integer32,
    lispEidRegistrationLastRegisterSender LispAddressType,
    lispEidRegistrationAuthenticationErrors Counter64,
    lispEidRegistrationRlocsMismatch Counter64
}

lispEidRegistrationEidLength OBJECT-TYPE
    SYNTAX      Integer32 (5..39)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object is used to get the octet-length of
        lispEidRegistrationEid."
    ::= { lispEidRegistrationEntry 1 }

lispEidRegistrationEid OBJECT-TYPE
    SYNTAX      LispAddressType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The EID prefix that is being registered."
    ::= { lispEidRegistrationEntry 2 }

lispEidRegistrationSiteName OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(0..63))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Site name used by a Map-Server to distinguish different
        LISP sites that are registering with it."
    ::= { lispEidRegistrationEntry 3 }

lispEidRegistrationSiteDescription OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(0..255))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Description for a site name used by a Map-Server. The EID
        prefix that is being registered belongs to this site."
    ::= { lispEidRegistrationEntry 4 }
```

```
lispEidRegistrationIsRegistered OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates the registration status of the given EID prefix.
        If this object is true, then it means the EID prefix is
        registered.

        The value false implies the EID prefix is not registered
        with the Map Server. There are multiple scenarios when this
        could happen like authentication failures, routing problems,
        misconfigs to name a few."
    ::= { lispEidRegistrationEntry 5 }

lispEidRegistrationFirstTimeStamp OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The value of sysUpTime at which the first valid register
        message for the EID Prefix information represented by this
        entry was received by this device.

        If this information was present at the most recent
        re-initialization of the local management subsystem, then
        this object contains a zero value."
    DEFVAL { 0 }
    ::= { lispEidRegistrationEntry 6 }

lispEidRegistrationLastTimeStamp OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The value of sysUpTime at which the last valid register
        message for the EID Prefix information represented by this
        entry was received by this device.

        If this information was present at the most recent
        re-initialization of the local management subsystem, then
        this object contains a zero value."
    DEFVAL { 0 }
    ::= { lispEidRegistrationEntry 7 }

lispEidRegistrationLastRegisterSenderLength OBJECT-TYPE
    SYNTAX      Integer32 (5..39)
    MAX-ACCESS  read-only
```

```
STATUS      current
DESCRIPTION
    "This object is used to get the octet-length of
    lispEidRegistrationLastRegisterSender, the next
    object."
 ::= { lispEidRegistrationEntry 8 }

lispEidRegistrationLastRegisterSender OBJECT-TYPE
SYNTAX      LispAddressType
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Source address of the last valid register message for the
    given EID prefix that was received by this device."
 ::= { lispEidRegistrationEntry 9 }

lispEidRegistrationAuthenticationErrors OBJECT-TYPE
SYNTAX      Counter64
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Count of total authentication errors of map-registers
    received for the given EID prefix.

    Discontinuities in this monotonically increasing value occur
    at re-initialization of the management system.
    Discontinuities can also occur as a result of site config
    changes, which can be detected by observing the value of
    lispEidRegistrationFirstTimeStamp."
 ::= { lispEidRegistrationEntry 10 }

lispEidRegistrationRlocsMismatch OBJECT-TYPE
SYNTAX      Counter64
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Count of total map-registers received that had at least one
    RLOC that was not in the allowed list of RLOCs for the given
    EID prefix.

    Discontinuities in this monotonically increasing value occur
    at re-initialization of the management system.
    Discontinuities can also occur as a result of site config
    changes, which can be detected by observing the value of
    lispEidRegistrationFirstTimeStamp."
 ::= { lispEidRegistrationEntry 11 }
```



```
lispEidRegistrationEtrTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LispEidRegistrationEtrEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table provides the properties of ETRs that register
        the given EID prefix with this device when configured to
        be a Map-Server."
    REFERENCE
        "RFC6830, Section 6.1."
    ::= { lispObjects 10 }

lispEidRegistrationEtrEntry OBJECT-TYPE
    SYNTAX      LispEidRegistrationEtrEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry (conceptual row) in the
        lispEidRegistrationEtrTable."
    INDEX       { lispEidRegistrationEidLength,
                  lispEidRegistrationEid,
                  lispEidRegistrationEtrSenderLength,
                  lispEidRegistrationEtrSender }
    ::= { lispEidRegistrationEtrTable 1 }

LispEidRegistrationEtrEntry ::= SEQUENCE {
    lispEidRegistrationEtrSenderLength      Integer32,
    lispEidRegistrationEtrSender            LispAddressType,
    lispEidRegistrationEtrLastTimeStamp     TimeStamp,
    lispEidRegistrationEtrTtl               Unsigned32,
    lispEidRegistrationEtrProxyReply        TruthValue,
    lispEidRegistrationEtrWantsMapNotify    TruthValue
}

lispEidRegistrationEtrSenderLength OBJECT-TYPE
    SYNTAX      Integer32 (5..39)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object is used to get the octet-length of
        lispEidRegistrationEtrSender."
    ::= { lispEidRegistrationEtrEntry 1 }

lispEidRegistrationEtrSender OBJECT-TYPE
    SYNTAX      LispAddressType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
```

```
        "Source address of the ETR that is sending valid register
        messages for this EID prefix to this device."
 ::= { lispEidRegistrationEtrEntry 2 }

lispEidRegistrationEtrLastTimeStamp OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The value of sysUpTime at which the last valid register
        message from this ETR for the EID Prefix information
        represented by this entry was received by this device.

        If this information was present at the most recent
        re-initialization of the local management subsystem,
        then this object contains a zero value."
    DEFVAL { 0 }
 ::= { lispEidRegistrationEtrEntry 3 }

lispEidRegistrationEtrTtl OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The Record TTL of the registering ETR device for this
        EID prefix."
 ::= { lispEidRegistrationEtrEntry 4 }

lispEidRegistrationEtrProxyReply OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates proxy-replying status of the registering ETR for
        this EID prefix. If this object is true, then it means the
        Map-Server can proxy-reply."
 ::= { lispEidRegistrationEtrEntry 5 }

lispEidRegistrationEtrWantsMapNotify OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates whether the EID prefix wants Map-Notifications.
        If this object is true, then it means the EID prefix wants
        Map-Notifications."
 ::= { lispEidRegistrationEtrEntry 6 }
```

```

lispEidRegistrationLocatorTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LispEidRegistrationLocatorEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table provides the properties of all locators per
        LISP site that are served by this device when configured
        to be a Map-Server."
    REFERENCE
        "RFC6830, Section 6.1."
    ::= { lispObjects 11 }

```

```

lispEidRegistrationLocatorEntry OBJECT-TYPE
    SYNTAX      LispEidRegistrationLocatorEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry (conceptual row) in the
        lispEidRegistrationLocatorTable."
    INDEX       { lispEidRegistrationEidLength,
                  lispEidRegistrationEid,
                  lispEidRegistrationEtrSenderLength,
                  lispEidRegistrationEtrSender,
                  lispEidRegistrationLocatorRlocLength,
                  lispEidRegistrationLocatorRloc }
    ::= { lispEidRegistrationLocatorTable 1 }

```

```

LispEidRegistrationLocatorEntry ::= SEQUENCE {
    lispEidRegistrationLocatorRlocLength  Integer32,
    lispEidRegistrationLocatorRloc        LispAddressType,
    lispEidRegistrationLocatorRlocState   INTEGER,
    lispEidRegistrationLocatorIsLocal     TruthValue,
    lispEidRegistrationLocatorPriority     Integer32,
    lispEidRegistrationLocatorWeight      Integer32,
    lispEidRegistrationLocatorMPriority   Integer32,
    lispEidRegistrationLocatorMWeight     Integer32
}

```

```

lispEidRegistrationLocatorRlocLength OBJECT-TYPE
    SYNTAX      Integer32 (5..39)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object is used to get the octet-length of
        lispEidRegistrationLocatorRloc."
    ::= { lispEidRegistrationLocatorEntry 1 }

```

```

lispEidRegistrationLocatorRloc OBJECT-TYPE

```

```
SYNTAX      LispAddressType
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The locator of the given EID prefix being registered by the
    given ETR with this device."
 ::= { lispEidRegistrationLocatorEntry 2 }

lispEidRegistrationLocatorRlocState OBJECT-TYPE
    SYNTAX      INTEGER {
                    up (1),
                    down (2)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The cached state of this RLOC received in map-register from
        the ETR by the device, in the capacity of a Map-Server.
        Value 1 refers to up, value 2 refers to down."
    ::= { lispEidRegistrationLocatorEntry 3 }

lispEidRegistrationLocatorIsLocal OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates if the given locator is local to the registering
        ETR. If this object is true, it means the locator is local."
    ::= { lispEidRegistrationLocatorEntry 4 }

lispEidRegistrationLocatorPriority OBJECT-TYPE
    SYNTAX      Integer32 (0..255)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The unicast priority of the RLOC for this EID prefix in the
        register message sent by the given ETR."
    ::= { lispEidRegistrationLocatorEntry 5 }

lispEidRegistrationLocatorWeight OBJECT-TYPE
    SYNTAX      Integer32 (0..100)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The unicast weight of the RLOC for this EID prefix in the
        register message sent by the given ETR."
    ::= { lispEidRegistrationLocatorEntry 6 }
```

```
lispEidRegistrationLocatorMPriority OBJECT-TYPE
    SYNTAX      Integer32 (0..255)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The multicast priority of the RLOC for this EID prefix in
        the register message sent by the given ETR."
    ::= { lispEidRegistrationLocatorEntry 7 }

lispEidRegistrationLocatorMWeight OBJECT-TYPE
    SYNTAX      Integer32 (0..100)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The multicast weight of the RLOC for this EID prefix in the
        register message sent by the given ETR."
    ::= { lispEidRegistrationLocatorEntry 8 }

lispUseMapServerTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LispUseMapServerEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table provides the properties of the map-server(s)
        with which this device is configured to register."
    REFERENCE
        "RFC6833, Section 4.3."
    ::= { lispObjects 12 }

lispUseMapServerEntry OBJECT-TYPE
    SYNTAX      LispUseMapServerEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry (conceptual row) in the lispUseMapServerTable."
    INDEX      { lispUseMapServerAddressLength,
                 lispUseMapServerAddress }
    ::= { lispUseMapServerTable 1 }

LispUseMapServerEntry ::= SEQUENCE {
    lispUseMapServerAddressLength Integer32,
    lispUseMapServerAddress      LispAddressType,
    lispUseMapServerState        INTEGER
}

lispUseMapServerAddressLength OBJECT-TYPE
    SYNTAX      Integer32 (5..39)
```

```
MAX-ACCESS not-accessible
STATUS      current
DESCRIPTION
    "This object is used to get the octet-length of
    lispUseMapServerAddress."
 ::= { lispUseMapServerEntry 1 }

lispUseMapServerAddress OBJECT-TYPE
SYNTAX      LispAddressType
MAX-ACCESS not-accessible
STATUS      current
DESCRIPTION
    "Address of Map-Server configured on this device."
 ::= { lispUseMapServerEntry 2 }

lispUseMapServerState OBJECT-TYPE
SYNTAX      INTEGER {
                up (1),
                down (2),
                unreachable (3)
            }
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
    "State of this Map-Server configured on this device
    (1 = Map-Server is up; 2 = Map-Server is down)."
 ::= { lispUseMapServerEntry 3 }

lispUseMapResolverTable OBJECT-TYPE
SYNTAX      SEQUENCE OF LispUseMapResolverEntry
MAX-ACCESS not-accessible
STATUS      current
DESCRIPTION
    "This table provides the properties of the map-resolver(s)
    this device is configured to use."
REFERENCE
    "RFC6833, Section 4.4."
 ::= { lispObjects 13 }

lispUseMapResolverEntry OBJECT-TYPE
SYNTAX      LispUseMapResolverEntry
MAX-ACCESS not-accessible
STATUS      current
DESCRIPTION
    "An entry (conceptual row) in the
    lispUseMapResolverTable."
```

```
INDEX      { lispUseMapResolverAddressLength,
              lispUseMapResolverAddress }
 ::= { lispUseMapResolverTable 1 }

LispUseMapResolverEntry ::= SEQUENCE {
    lispUseMapResolverAddressLength  Integer32,
    lispUseMapResolverAddress        LispAddressType,
    lispUseMapResolverState          INTEGER
}

lispUseMapResolverAddressLength OBJECT-TYPE
    SYNTAX      Integer32 (5..39)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object is used to get the octet-length of
        lispUseMapResolverAddress."
    ::= { lispUseMapResolverEntry 1 }

lispUseMapResolverAddress OBJECT-TYPE
    SYNTAX      LispAddressType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Address of map-resolver configured on this device."
    ::= { lispUseMapResolverEntry 2 }

lispUseMapResolverState OBJECT-TYPE
    SYNTAX      INTEGER {
        up (1),
        down (2)
    }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "State of this Map-Resolver configured on this device
        (1 = Map-Resolver is up; 2 = Map-Resolver is down)."
    ::= { lispUseMapResolverEntry 3 }

lispUseProxyEtrTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LispUseProxyEtrEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table provides the properties of all Proxy ETRs that
        this device is configured to use."
```

## REFERENCE

"RFC6830, Section 6.0."

::= { lispObjects 14 }

## lispUseProxyEtrEntry OBJECT-TYPE

SYNTAX LispUseProxyEtrEntry

MAX-ACCESS not-accessible

STATUS current

## DESCRIPTION

"An entry (conceptual row) in the  
lispUseProxyEtrTable."

INDEX { lispUseProxyEtrAddressLength,  
lispUseProxyEtrAddress }

::= { lispUseProxyEtrTable 1 }

## LispUseProxyEtrEntry ::= SEQUENCE {

lispUseProxyEtrAddressLength	Integer32,
lispUseProxyEtrAddress	LispAddressType,
lispUseProxyEtrPriority	Integer32,
lispUseProxyEtrWeight	Integer32,
lispUseProxyEtrMPriority	Integer32,
lispUseProxyEtrMWeight	Integer32,
lispUseProxyEtrState	INTEGER

}

## lispUseProxyEtrAddressLength OBJECT-TYPE

SYNTAX Integer32 (5..39)

MAX-ACCESS not-accessible

STATUS current

## DESCRIPTION

"This object is used to get the octet-length of  
lispUseProxyEtrAddress."

::= { lispUseProxyEtrEntry 1 }

## lispUseProxyEtrAddress OBJECT-TYPE

SYNTAX LispAddressType

MAX-ACCESS not-accessible

STATUS current

## DESCRIPTION

"Address of Proxy ETR configured on this device."

::= { lispUseProxyEtrEntry 2 }

## lispUseProxyEtrPriority OBJECT-TYPE

SYNTAX Integer32 (0..255)

MAX-ACCESS read-only

STATUS current

## DESCRIPTION

"The unicast priority of the PETR locator."



```
 ::= { lispUseProxyEtrEntry 3 }

lispUseProxyEtrWeight OBJECT-TYPE
    SYNTAX      Integer32 (0..100)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The unicast weight of the PETR locator."
    ::= { lispUseProxyEtrEntry 4 }

lispUseProxyEtrMPriority OBJECT-TYPE
    SYNTAX      Integer32 (0..255)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The multicast priority of the PETR locator."
    ::= { lispUseProxyEtrEntry 5 }

lispUseProxyEtrMWeight OBJECT-TYPE
    SYNTAX      Integer32 (0..100)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The multicast weight of the PETR locator."
    ::= { lispUseProxyEtrEntry 6 }

lispUseProxyEtrState OBJECT-TYPE
    SYNTAX      INTEGER {
                    down (0),
                    up (1)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "State of this Proxy ETR configured on this device
        (0 = Proxy ETR is down; 1 = Proxy ETR is up)."
    ::= { lispUseProxyEtrEntry 7 }
```

```
--
-- Conformance Information
--

lispCompliances OBJECT IDENTIFIER ::= { lispConformance 1 }
lispGroups       OBJECT IDENTIFIER ::= { lispConformance 2 }


--
-- Compliance Statements
--

lispMIBComplianceEtr MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
        "The compliance statement for LISP ETRs. It conveys
        information if device supports ETR feature, and relevant
        state associated with that feature."
    MODULE      -- this module
    MANDATORY-GROUPS { lispMIBetrGroup }

    GROUP      lispMIBitrGroup
    DESCRIPTION
        "This group is optional."

    GROUP      lispMIBPetrGroup
    DESCRIPTION
        "This group is optional."

    GROUP      lispMIBPitrGroup
    DESCRIPTION
        "This group is optional."

    GROUP      lispMIBMapServerGroup
    DESCRIPTION
        "This group is optional."

    GROUP      lispMIBMapResolverGroup
    DESCRIPTION
        "This group is optional."

    GROUP      lispMIBetrExtendedGroup
    DESCRIPTION
        "This group is optional."

    GROUP      lispMIBitrExtendedGroup
    DESCRIPTION
        "This group is optional."
```

```
GROUP    lispMIBMapServerExtendedGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBTuningParametersGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBEncapStatisticsGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBDecapStatisticsGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBDiagnosticsGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBVrfGroup
DESCRIPTION
    "This group is optional."

 ::= { lispCompliances 1 }

lispMIBComplianceItr MODULE-COMPLIANCE
STATUS    current
DESCRIPTION
    "The compliance statement for LISP ITRs. It conveys
    information if device supports ITR feature, and any
    state associated with that feature."
MODULE    -- this module
MANDATORY-GROUPS { lispMIBItrGroup }

GROUP    lispMIBEtrGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBPetrGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBPitrGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBMapServerGroup
```

```
DESCRIPTION
    "This group is optional."

GROUP      lispMIBMapResolverGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBEtrExtendedGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBItrExtendedGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBMapServerExtendedGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBTuningParametersGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBEncapStatisticsGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBDecapStatisticsGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBDiagnosticsGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBVrfGroup
DESCRIPTION
    "This group is optional."

 ::= { lispCompliances 2 }

lispMIBCompliancePetr MODULE-COMPLIANCE
STATUS      current
DESCRIPTION
    "The compliance statement for LISP Proxy-ETRs. It conveys
    information if given device supports Proxy-ETR feature,
    and relevant state associated with that feature."
MODULE      -- this module
```

```
MANDATORY-GROUPS { lispMIBPetrGroup }

GROUP      lispMIBEtrGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBItrGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBPitrGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBMapServerGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBMapResolverGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBEtrExtendedGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBItrExtendedGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBMapServerExtendedGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBTuningParametersGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBEncapStatisticsGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBDecapStatisticsGroup
DESCRIPTION
    "This group is optional."

GROUP      lispMIBDiagnosticsGroup
DESCRIPTION
```

```
        "This group is optional."

    GROUP    lispMIBVrfGroup
    DESCRIPTION
        "This group is optional."

 ::= { lispCompliances 3 }

lispMIBCompliancePitr MODULE-COMPLIANCE
    STATUS    current
    DESCRIPTION
        "The compliance statement for LISP Proxy-ITRs. It conveys
        information if device supports Proxy-ITR feature, and
        relevant state associated with that feature."
    MODULE    -- this module
    MANDATORY-GROUPS { lispMIBPitrGroup }

    GROUP    lispMIBEtrGroup
    DESCRIPTION
        "This group is optional."

    GROUP    lispMIBItrGroup
    DESCRIPTION
        "This group is optional."

    GROUP    lispMIBPetrGroup
    DESCRIPTION
        "This group is optional."

    GROUP    lispMIBMapServerGroup
    DESCRIPTION
        "This group is optional."

    GROUP    lispMIBMapResolverGroup
    DESCRIPTION
        "This group is optional."

    GROUP    lispMIBEtrExtendedGroup
    DESCRIPTION
        "This group is optional."

    GROUP    lispMIBItrExtendedGroup
    DESCRIPTION
        "This group is optional."

    GROUP    lispMIBMapServerExtendedGroup
    DESCRIPTION
        "This group is optional."
```

```
GROUP    lispMIBTuningParametersGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBEncapStatisticsGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBDecapStatisticsGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBDiagnosticsGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBVrfGroup
DESCRIPTION
    "This group is optional."

 ::= { lispCompliances 4 }

lispMIBComplianceMapServer MODULE-COMPLIANCE
STATUS    current
DESCRIPTION
    "The compliance statement for LISP Map Servers. It
    conveys information if device supports Map Server
    feature, and relevant state associated with that
    feature."
MODULE    -- this module
MANDATORY-GROUPS { lispMIBMapServerGroup }

GROUP    lispMIBEtrGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBItrGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBPetrGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBPitrGroup
DESCRIPTION
    "This group is optional."
```

```
GROUP    lispMIBMapResolverGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBEtrExtendedGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBItrExtendedGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBMapServerExtendedGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBTuningParametersGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBEncapStatisticsGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBDecapStatisticsGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBDiagnosticsGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBVrfGroup
DESCRIPTION
    "This group is optional."

 ::= { lispCompliances 5 }

lispMIBComplianceMapResolver MODULE-COMPLIANCE
STATUS    current
DESCRIPTION
    "The compliance statement for LISP Map Resolvers. It
    conveys information if device supports Map Server
    feature, and relevant state associated with that
    feature."
MODULE    -- this module
MANDATORY-GROUPS { lispMIBMapResolverGroup }
```



```
GROUP    lispMIBetrGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBitrGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBPetrGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBPitrGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBMapServerGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBetrExtendedGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBitrExtendedGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBMapServerExtendedGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBTuningParametersGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBEncapStatisticsGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBDecapStatisticsGroup
DESCRIPTION
    "This group is optional."

GROUP    lispMIBDiagnosticsGroup
DESCRIPTION
    "This group is optional."
```

```
GROUP    lispMIBVrfGroup
DESCRIPTION
    "This group is optional."

 ::= { lispCompliances 6 }

--
-- Units of Conformance
--

lispMIBetrGroup OBJECT-GROUP
    OBJECTS { lispFeaturesEtrEnabled,
               lispMappingDatabaseLsb,
               lispMappingDatabaseLocatorRlocPriority,
               lispMappingDatabaseLocatorRlocWeight,
               lispMappingDatabaseLocatorRlocMPriority,
               lispMappingDatabaseLocatorRlocMWeight,
               lispMappingDatabaseLocatorRlocState,
               lispMappingDatabaseLocatorRlocLocal,
               lispConfiguredLocatorRlocState,
               lispConfiguredLocatorRlocLocal,
               lispUseMapServerState
             }
    STATUS current
    DESCRIPTION
        "A collection of objects to support reporting of basic
         LISP ETR parameters."
    ::= { lispGroups 1 }

lispMIBitrGroup OBJECT-GROUP
    OBJECTS { lispFeaturesItrEnabled,
               lispFeaturesMapCacheSize,
               lispMappingDatabaseLsb,
               lispMapCacheLocatorRlocPriority,
               lispMapCacheLocatorRlocWeight,
               lispMapCacheLocatorRlocMPriority,
               lispMapCacheLocatorRlocMWeight,
               lispMapCacheLocatorRlocState,
               lispMapCacheEidTimeStamp,
               lispMapCacheEidExpiryTime,
               lispUseMapResolverState,
               lispUseProxyEtrPriority,
               lispUseProxyEtrWeight,
               lispUseProxyEtrMPriority,
               lispUseProxyEtrMWeight,
               lispUseProxyEtrState
             }
    }
```

```
STATUS    current
DESCRIPTION
    "A collection of objects to support reporting of basic
    LISP ITR parameters."
 ::= { lispGroups 2 }

lispMIBPetrGroup OBJECT-GROUP
    OBJECTS { lispFeaturesProxyEtrEnabled
    }
    STATUS    current
    DESCRIPTION
        "A collection of objects to support reporting of basic
        LISP Proxy-ETR parameters."
    ::= { lispGroups 3 }

lispMIBPitrGroup OBJECT-GROUP
    OBJECTS { lispFeaturesProxyItrEnabled,
               lispConfiguredLocatorRlocState,
               lispConfiguredLocatorRlocLocal
    }

    STATUS    current
    DESCRIPTION
        "A collection of objects to support reporting of basic
        LISP Proxy-ITR parameters."
    ::= { lispGroups 4 }

lispMIBMapServerGroup OBJECT-GROUP
    OBJECTS { lispFeaturesMapServerEnabled,
               lispEidRegistrationIsRegistered,
               lispEidRegistrationLocatorRlocState
    }
    STATUS    current
    DESCRIPTION
        "A collection of objects to support reporting of basic
        LISP Map Server parameters."
    ::= { lispGroups 5 }

lispMIBMapResolverGroup OBJECT-GROUP
    OBJECTS { lispFeaturesMapResolverEnabled
    }
    STATUS    current
    DESCRIPTION
        "A collection of objects to support reporting of basic
        LISP Map Resolver parameters."
    ::= { lispGroups 6 }

lispMIBEtrExtendedGroup OBJECT-GROUP
```

```
OBJECTS { lispFeaturesRlocProbeEnabled,
          lispFeaturesEtrAcceptMapDataEnabled,
          lispFeaturesEtrAcceptMapDataVerifyEnabled,
          lispMappingDatabaseEidPartitioned
        }
STATUS   current
DESCRIPTION
    "A collection of objects to support reporting of
    LISP features and properties on ETRs."
 ::= { lispGroups 7 }

lispMIBItrExtendedGroup OBJECT-GROUP
OBJECTS { lispFeaturesRlocProbeEnabled,
          lispMapCacheEidState,
          lispMapCacheEidAuthoritative,
          lispMapCacheLocatorRlocTimeStamp,
          lispMapCacheLocatorRlocLastPriorityChange,
          lispMapCacheLocatorRlocLastWeightChange,
          lispMapCacheLocatorRlocLastMPriorityChange,
          lispMapCacheLocatorRlocLastMWeightChange,
          lispMapCacheLocatorRlocLastStateChange,
          lispMapCacheLocatorRlocRtt
        }
STATUS   current
DESCRIPTION
    "A collection of objects to support reporting of
    LISP features and properties on ITRs."
 ::= { lispGroups 8 }

lispMIBMapServerExtendedGroup OBJECT-GROUP
OBJECTS { lispEidRegistrationSiteName,
          lispEidRegistrationSiteDescription,
          lispEidRegistrationIsRegistered,
          lispEidRegistrationFirstTimeStamp,
          lispEidRegistrationLastTimeStamp,
          lispEidRegistrationLastRegisterSenderLength,
          lispEidRegistrationLastRegisterSender,
          lispEidRegistrationEtrLastTimeStamp,
          lispEidRegistrationEtrTtl,
          lispEidRegistrationEtrProxyReply,
          lispEidRegistrationEtrWantsMapNotify,
          lispEidRegistrationLocatorIsLocal,
          lispEidRegistrationLocatorPriority,
          lispEidRegistrationLocatorWeight,
          lispEidRegistrationLocatorMPriority,
          lispEidRegistrationLocatorMWeight
        }
STATUS   current
```

```
DESCRIPTION
    "A collection of objects to support reporting of
      LISP features and properties on Map Servers
      related to EID registrations."
 ::= { lispGroups 9 }

lispMIBTuningParametersGroup OBJECT-GROUP
  OBJECTS { lispFeaturesMapCacheLimit,
            lispFeaturesEtrMapCacheTtl
          }
  STATUS   current
  DESCRIPTION
    "A collection of objects used to support reporting of
      parameters used to control LISP behavior and to tune
      performance."
 ::= { lispGroups 10 }

lispMIBEncapStatisticsGroup OBJECT-GROUP
  OBJECTS { lispMappingDatabaseTimeStamp,
            lispMappingDatabaseEncapOctets,
            lispMappingDatabaseEncapPackets,
            lispMappingDatabaseLocatorRlocTimeStamp,
            lispMappingDatabaseLocatorRlocEncapOctets,
            lispMappingDatabaseLocatorRlocEncapPackets,
            lispMapCacheEidTimeStamp,
            lispMapCacheEidEncapOctets,
            lispMapCacheEidEncapPackets,
            lispMapCacheLocatorRlocTimeStamp,
            lispMapCacheLocatorRlocEncapOctets,
            lispMapCacheLocatorRlocEncapPackets,
            lispConfiguredLocatorRlocTimeStamp,
            lispConfiguredLocatorRlocEncapOctets,
            lispConfiguredLocatorRlocEncapPackets
          }
  STATUS   current
  DESCRIPTION
    "A collection of objects used to support reporting of
      LISP encapsulation statistics for the device."
 ::= { lispGroups 11 }

lispMIBDecapStatisticsGroup OBJECT-GROUP
  OBJECTS { lispMappingDatabaseTimeStamp,
            lispMappingDatabaseDecapOctets,
            lispMappingDatabaseDecapPackets,
            lispMappingDatabaseLocatorRlocTimeStamp,
            lispMappingDatabaseLocatorRlocDecapOctets,
            lispMappingDatabaseLocatorRlocDecapPackets,
            lispMapCacheEidTimeStamp,
```

```
        lispMapCacheEidDecapOctets,
        lispMapCacheEidDecapPackets,
        lispMapCacheLocatorRlocTimeStamp,
        lispMapCacheLocatorRlocDecapOctets,
        lispMapCacheLocatorRlocDecapPackets,
        lispConfiguredLocatorRlocTimeStamp,
        lispConfiguredLocatorRlocDecapOctets,
        lispConfiguredLocatorRlocDecapPackets
    }
    STATUS    current
    DESCRIPTION
        "A collection of objects used to support reporting of
        LISP decapsulation statistics for the device."
    ::= { lispGroups 12 }

lispMIBDiagnosticsGroup OBJECT-GROUP
    OBJECTS { lispFeaturesRouterTimeStamp,
              lispGlobalStatsMapRequestsIn,
              lispGlobalStatsMapRequestsOut,
              lispGlobalStatsMapRepliesIn,
              lispGlobalStatsMapRepliesOut,
              lispGlobalStatsMapRegistersIn,
              lispGlobalStatsMapRegistersOut,
              lispEidRegistrationAuthenticationErrors,
              lispEidRegistrationRlocsMismatch
    }
    STATUS    current
    DESCRIPTION
        "A collection of objects used to support reporting of
        additional diagnostics related to the LISP control plane
        state of a LISP device."
    ::= { lispGroups 13 }

lispMIBVrfGroup OBJECT-GROUP
    OBJECTS { lispIidToVrfName
    }
    STATUS    current
    DESCRIPTION
        "A collection of objects used to support reporting of
        VRF-related information on a LISP device."
    ::= { lispGroups 14 }

END
```

## 8. Relationship to Other MIB Modules

### 8.1. MIB modules required for IMPORTS

The LISP MIB imports the TEXTUAL-CONVENTION AddressFamilyNumbers from the IANA-ADDRESS-FAMILY-NUMBERS-MIB DEFINITIONS [IANA]  
<http://www.iana.org/assignments/ianaaddressfamilynumbers-mib>

The LISP MIB imports mib-2, Unsigned32, Counter64, Integer32, and TimeTicks from SNMPv2-SMI -- [RFC2578].

The LISP MIB imports TruthValue, TEXTUAL-CONVENTION, TimeStamp, and TimeTicks from SNMPv2-TC -- [RFC2579].

The LISP MIB imports MODULE-COMPLIANCE from SNMPv2-TC -- [RFC2580].

The LISP MIB imports MplsL3VpnName from MPLS-L3VPN-STD-MIB -- [RFC4382].

## 9. Security Considerations

There are no management objects defined in this MIB module that have a MAX-ACCESS clause of read-write and/or read-create. So, if this MIB module is implemented correctly, then there is no risk that an intruder can alter or create any management objects of this MIB module via direct SNMP SET operations.

There are no readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) that are considered sensitive.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

Implementations SHOULD provide the security features described by the SNMPv3 framework (see [RFC3410]), and implementations claiming compliance to the SNMPv3 standard MUST include full support for authentication and privacy via the User-based Security Model (USM) [RFC3414] with the AES cipher algorithm [RFC3826]. Implementations MAY also provide support for the Transport Security Model (TSM) [RFC5591] in combination with a secure transport such as SSH [RFC5592] or TLS/DTLS [RFC6353].

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to

enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

## 10. IANA Considerations

The MIB module in this document uses the following IANA-assigned OBJECT IDENTIFIER values recorded in the SMI Numbers registry:

Descriptor -----	OBJECT IDENTIFIER value -----
lispMIB	{ mib-2 XXX }

This document instructs IANA to allocate a new value in the "SMI Network Management MGMT Codes Internet-standard MIB" subregistry of the "Network Management Parameters" registry, according to the following registration data: Decimal: [TBD by IANA] Name: lispMIB Description: Locator/ID Separation Protocol (LISP) References: [RFC XXXX (this RFC)]

## 11. References

### 11.1. Normative References

- [IANA] "IANA-ADDRESS-FAMILY-NUMBERS-MIB DEFINITIONS", <<http://www.iana.org/assignments/ianaaddressfamilynumbers-mib>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.
- [RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.
- [RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.
- [RFC3414] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management



Protocol (SNMPv3)", STD 62, RFC 3414, December 2002.

- [RFC3826] Blumenthal, U., Maino, F., and K. McCloghrie, "The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model", RFC 3826, June 2004.
- [RFC4382] Nadeau, T. and H. van der Linde, "MPLS/BGP Layer 3 Virtual Private Network (VPN) Management Information Base", RFC 4382, February 2006.
- [RFC5591] Harrington, D. and W. Hardaker, "Transport Security Model for the Simple Network Management Protocol (SNMP)", RFC 5591, June 2009.
- [RFC5592] Harrington, D., Salowey, J., and W. Hardaker, "Secure Shell Transport Model for the Simple Network Management Protocol (SNMP)", RFC 5592, June 2009.
- [RFC6353] Hardaker, W., "Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)", RFC 6353, July 2011.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.
- [RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, January 2013.
- [RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, January 2013.

#### 11.2. Informative References

- [LCAF] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format", draft-ietf-lisp-lcaf-02.txt (work in progress), March 2013.
- [RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

#### Appendix A. Acknowledgments

A thank you is owed to Dino Farinacci for his inputs and review comments on the initial versions of this draft. In addition, the

authors would like to gratefully acknowledge several others who have reviewed and commented on this draft. They include: Darrel Lewis, Isidor Kouvelas, Jesper Skriver, Selina Heimlich, Parna Agrawal, Dan Romascanu, and Luigi Iannone. Special thanks are owed to Brian Haberman, the Internet Area AD, for his very detailed review, Miguel Garcia for reviewing this document as part of the General Area Review Team, and Harrie Hazewinkel for the detailed MIB review comments.

#### Authors' Addresses

Gregg Schudel  
cisco Systems  
Tasman Drive  
San Jose, CA 95134  
USA

EMail: gschudel@cisco.com

Amit Jain  
Juniper Networks  
1133 Innovation Way  
Sunnyvale, CA 94089  
USA

EMail: atjain@juniper.net

Victor Moreno  
cisco Systems  
Tasman Drive  
San Jose, CA 95134  
USA

EMail: vimoreno@cisco.com



Network Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: January 2, 2013

D. Saucez  
INRIA  
O. Bonaventure  
UCLouvain  
L. Iannone  
Telecom ParisTech  
C. Filsfils  
Cisco Systems  
July 1, 2012

LISP ITR Graceful Restart  
draft-saucez-lisp-itr-graceful-00.txt

Abstract

The Locator/ID Separation Protocol (LISP) is a map-and-encap mechanism to enable the communication between hosts identified with their Endpoint Identifier (EID) over the Internet where EIDs are not routable. To do so, packets toward EIDs are encapsulated in packets with routing locators (RLOCs) to form dynamic tunnels. An Ingress Tunnel Router (ITR) that encapsulates EID packets determines tunnel endpoints via mappings that associate EIDs to RLOCs. Before encapsulating a packet, the ITR queries the mapping system to obtain the mapping associated to the EID of the packet it must encapsulate. Such mapping is cached by the ITR in its local EID-to-RLOC cache for any subsequent encapsulation for the same EID. LISP is scalable because the EID-to-RLOC cache of an ITR, which is initially empty, is populated progressively according to the traffic going through the ITR. However, after an ITR is restarted, e.g., for maintenance reason, its cache is empty which means that all packets that are re-routed to the freshly restarted ITR will cause cache misses and a potentially high loss rate. In this draft, we present mechanisms to reduce the negative impact on traffic caused by the restart of an ITR in a LISP network.

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 <http://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 January 2, 2013.

#### Copyright Notice

Copyright (c) 2012 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 (<http://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 . . . . .	4
2. Definition of terms . . . . .	4
2.1. LISP Definition of Terms . . . . .	5
3. Problem Statement . . . . .	7
4. ITR Graceful Restart . . . . .	8
5. Security Considerations . . . . .	9
6. Conclusion . . . . .	9
7. Acknowledgments . . . . .	9
8. References . . . . .	9
8.1. Normative References . . . . .	9
8.2. Informative References . . . . .	10
Authors' Addresses . . . . .	10

## 1. Introduction

The Locator/ID Separation Protocol (LISP) [I-D.ietf-lisp] relies on two principles. First, Endpoint Identifiers (EIDs) are allocated to hosts while Routing Locators (RLOCs) are allocated to LISP Ingress Tunnel Routers (ITR) and Egress Tunnel Routers (ETR). The EIDs are not directly routable on the global Internet, only the RLOCs are. Second, LISP relies on mapping and encapsulation. Hosts are located on sites and are served by ITRs and ETRs. When host A.1 in site A needs to send a packet to host B.2 in site B, its packet is intercepted by the ITR that serves its site. This ITR queries a mapping system to find the RLOC of the ETR that serves EID B.2. Once the RLOC of the ETR serving B's site is known, the ITR encapsulates the packet using the encapsulation defined in [I-D.ietf-lisp] so that it can reach B's ETR. B's ETR decapsulates the packet and forwards it to host B.

Packets from a LISP site are routed to their closest ITR by the mean of the routing system (e.g., IGP). In case of an ITR that just booted (either because it has just been added to the network or because it has been restarted due to maintenance) a large portion of the traffic can potentially be routed to the freshly started ITR. However, in this case, its EID-to-RLOC cache is empty. While with traditional routing, such a massive redirection has minor impact on the traffic (except for path stretch and latency), this can cause a high miss rate (i.e., no EID-to-RLOC Cache entry matching the destination RLOC) and hence packet loss. Such a miss storm includes a burst of Map-Requests that may overload the mapping system.

This memo aims at starting a discussion about ITR graceful (re)start in LISP networks. In this memo, we discuss the problem of ITR (re)start with the associated risk of miss storm and discuss EID-to-RLOC cache synchronization to provide ITR graceful restart without overwhelming the mapping system or packet loss.

## 2. Definition of terms

This section introduces the definition of the main elements and terms used throughout the whole document. More specifically, hereafter the terms introduced by this document are defined, while in Section 2.1 the definitions related to the LISP's architecture are provided in order to ease the read of the present document.

- o Cache Miss Storm: a period during which the observed cache miss rate at an ITR is significantly higher than the rate observed at the steady state on the ITR is called a Cache Miss Storm.

- o Synchronization Set: the set of ITRs that are potentially on the path of the same traffic should have their EID-to-RLOC cache synchronized in order to avoid Cache Miss Storms.
- o ITR restart: generic term indicating an ITR that has just completed the bootstrap phase and resuming normal operation. It can be either an ITR that has been added to the network (hence, actually at its first boot as part of the specific network) or an ITR actually re-booting due to reasons like, for instance, maintenance, temporary outage, etc.

## 2.1. LISP Definition of Terms

LISP operates on two name spaces and introduces several new network elements. This section provides high-level definitions of the LISP name spaces and network elements and as such, it MUST NOT be considered as an authoritative source. The reference to the authoritative document for each term is included in every term description.

- o Ingress Tunnel Router (ITR) [I-D.ietf-lisp]: 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 globally-routable RLOCs 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. Specifically, when a service provider prepends a LISP header for Traffic Engineering purposes, the router that does this is also regarded as an ITR. The outer RLOC the ISP ITR uses can be based on the outer destination address (the originating ITR's supplied RLOC) or the inner destination address (the originating hosts supplied EID).
- o Egress Tunnel Router (ETR) [I-D.ietf-lisp]: 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.
- o Routing Locator (RLOC) [I-D.ietf-lisp]: A RLOC is an IPv4 [RFC0791] or IPv6 [RFC2460] address of an egress tunnel router (ETR). A RLOC is the output of an EID-to-RLOC mapping lookup. An



EID maps to one or more RLOCs. Typically, RLOCs are numbered from topologically-aggregatable blocks that are assigned to a site at each point to which it attaches to the global Internet; where the topology is defined by the connectivity of provider networks, RLOCs can be thought of as PA addresses. Multiple RLOCs can be assigned to the same ETR device or to multiple ETR devices at a site.

- o Endpoint ID (EID) [I-D.ietf-lisp]: An EID is a 32-bit (for IPv4) or 128-bit (for IPv6) value used 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 an destination address today, for example through a Domain Name System (DNS) [RFC1034] lookup or Session Invitation 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 globally 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. EIDs MUST NOT be used as LISP RLOCs. 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 global routing system. In theory, the bit string that represents an EID for one device can represent an RLOC for a different device. As the architecture is realized, if a given bit string is both an RLOC and an EID, it must refer to the same entity in both cases. When used in discussions with other Locator/ID separation proposals, a LISP EID will be called a "LEID". Throughout this document, any references to "EID" refers to an LEID.
- o EID-to-RLOC Cache [I-D.ietf-lisp]: The EID-to-RLOC cache is a 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 global in scope.
- o EID-to-RLOC Database [I-D.ietf-lisp]: The EID-to-RLOC database is a global 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, globally-visible, IP addresses. The same database mapping entries MUST be configured on all ETRs for a given site. In a steady state the EID-prefixes for the site and the locator-set for

each EID-prefix MUST be the same on all ETRs. Procedures to enforce and/or verify this are outside the scope of this document. Note that there MAY be transient conditions when the EID-prefix for the 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.

### 3. Problem Statement

LISP is a map-and-encap mechanism where an ITR dynamically learns the mappings when it receives a packet for a destination EID for which it did not make encapsulation before. When such a packet is received, a cache miss occurs and the ITR sends a Map-Request to the mapping system to retrieve the mapping that corresponds to the destination that caused the cache miss. The ITR then caches the mapping for any subsequent packet toward the same destination. LISP [I-D.ietf-lisp] does not specify how a packet that causes a cache miss must be handled. However, to the best of our knowledge, the current implementations drop packets causing a cache miss. The impact of a miss is thus two-fold. On the one hand, misses imply packet losses and hence performance issues. On the other hand, due to the consequent Map-Request, cache misses cause load on the mapping system.

When an ITR restarts, its EID-to-RLOC cache is initially empty, and grows progressively with the traffic. However because mappings have a limited lifetime, the EID-to-RLOC cache size converges to a stable value and it is expected to always observe misses. As shown in [Networking12], at the steady state, networks experience a rather stable, and limited, miss rate. However, when an ITR is restarted, e.g., for a maintenance operation, a cache miss storm can be observed. A cache miss storm is a phenomenon during which the miss rate is significantly higher than the miss rate normally observed in the network. A miss storm has two severe side effects, first, it abruptly increases the load on the mapping system, and second, many packets are dropped, which causes performance issues. When an ITR is restarted, actually two cache miss storms can be observed. The first when the ITR is stopped (or fails) and, the second when the ITR is again available for encapsulation. The first miss storm is due to the fact that all the traffic is suddenly redirected to the other ITRs in the network, which might not have the mappings for all the EIDs of ongoing communications. The second miss storm can be observed when the ITR is restarted, because it might have to encapsulate all the traffic redirected to it. Indeed, when the ITR is freshly restarted, its cache is empty meaning that every packet will cause misses at that particular time.

Cache misses are normal in a LISP network. However, these misses normally happen only when the first packet of the first flow toward an EID is received by an ITR which have no significant impact on the traffic at steady state in the network. On the contrary, when an ITR restarts, cache misses happen on elapsing, potentially high throughput, flows for which high loss rate is not acceptable. For this particular reason, techniques must be applied to avoid miss storm upon ITRs restarts.

In this memo, we open the discussion on techniques that can be used to avoid cache miss storms in the case of a planned ITR restart. In other words, we discuss how to achieve ITR graceful restart.

#### 4. ITR Graceful Restart

The addition of an ITR causes the traffic to be redirected to the freshly started ITRs and hence risks to cause miss storm. Indeed, the cache of an ITR is empty when it starts so every packet received potentially causes a miss. We can isolate three techniques to protect the network from miss storm when an ITR is added (or restarted) in the network. All the ITRs that are potentially used by the same node in the network are grouped in synchronization sets.

- o Non-volatile mapping storage: when an ITR has to be stopped, its EID-to-RLOC Cache is stored on a non-volatile medium (e.g., a hard drive) such that when it is restarted, it can load the EID-to-RLOC cache to be equivalent of the cache it had before it restarted.
- o ITR deflection: when a miss occurs at an ITR while it is starting up, the ITR deflects the packet that caused a miss to an ITR in its synchronization set and, in parallel, sends a Map-Request for the EID that caused the miss.
- o ITR cache synchronization: upon startup, the ITR synchronizes its cache with the other ITRs in its synchronization set. The ITR is marked as available only after the cache is synchronized.

The non-volatile storage offers the advantage to be transparent for the network and is adapted to short unavailability periods (e.g., the ITR reboots after an upgrade). However, this technique is not adapted for long unavailability periods where most of the entries might be outdated and new prefixes unknown, or when an ITR is added for the first time in the network. This technique is thus recommended only for network with a low mapping caching dynamics.

Traffic deflection to other ITRs upon misses causes several issues. On the one hand, the ITR that is restarting must determine the ITR to which the packet must be deflected. On the other hand, packets must be marked as deflected in order to avoid loops. In addition, the ITR must determine its graceful restart period such that it stops

deflecting traffic once at steady state. The deflection from one ITR to another can be done directly in LISP where the ITR that started LISP encapsulates and forwards the packet to another ITR. This last ITR must then also run the ETR functionality to decapsulate the packet.

ITR cache synchronization is the most adapted to graceful restart. When the ITR starts, it sends requests to an ITR in its synchronization set (or its MR) to obtain the full cache. When the synchronization is finished, the ITR advertises itself as an ITR in the network such that the ITR does receive traffic to encapsulate only once its cache is synchronized.

## 5. Security Considerations

Security considerations have to be written accordingly to the technique finally chosen for ITR graceful restart. However, as a general security recommendation, we can say that the mappings must be authenticated in order to avoid relay attacks or denial of service. However, ITR graceful restart should not introduce any new threat in the core LISP mechanism.

## 6. Conclusion

In this memo, we highlighted the implication of the addition or the restart of an ITR in a LISP network. When an ITR is added into a LISP network, its EID-to-RLOC Cache is initially empty. Therefore, when on-going flows are routed to the freshly started ITR, their packets cause potential miss storm which results in packet drops and mapping system overload. To tackle this issue, we propose three different techniques to reduce the impact of a planned ITR restart.

## 7. Acknowledgments

The authors would like to acknowledge Dino Farinacci, Vince Fuller, Darrel Lewis, Fabio Maino, and Simon van der Linden.

## 8. References

### 8.1. Normative References

[I-D.ietf-lisp]  
Farinacci, D., Fuller, V., Meyer, D., and D. Lewis,  
"Locator/ID Separation Protocol (LISP)",

draft-ietf-lisp-22 (work in progress), February 2012.

## 8.2. Informative References

- [Networking12] Saucez, D., Kim, J., Iannone, L., Bonaventure, O., and C. Filsfils, "A local Approach to Fast Failure Recovery of LISP Ingress Tunnel Routers", The 11th International Conference on Networking (Networking'12) , May 2012, <[Networking12]>.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

## Authors' Addresses

Damien Saucez  
INRIA  
2004 route des Lucioles BP 93  
Sophia Antipolis Cedex, 06902  
France

Email: damien.saucez@inria.fr

Olivier Bonaventure  
UCLouvain  
Universite catholique de Louvain, Place Sainte Barbe 2  
Louvain-la-Neuve, 1348  
Belgium

Email: olivier.bonaventure@uclouvain.be  
URI: <http://inl.info.ucl.ac.be>

Luigi Iannone  
Telecom ParisTech  
23, Avenue d'Italie  
75013 Paris  
France

Email: [luigi.iannone@telecom-paristech.fr](mailto:luigi.iannone@telecom-paristech.fr)

Clarence Filsfils  
Cisco Systems  
Brussels, 1000  
Belgium

Email: [cf@cisco.com](mailto:cf@cisco.com)

