

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

F. Brockners
S. Bhandari
V. Govindan
C. Pignataro
Cisco
H. Gredler
RtBrick Inc.
J. Leddy
Comcast
S. Youell
JMPC
T. Mizrahi
Marvell
D. Mozes
Mellanox Technologies Ltd.
P. Lapukhov
Facebook
R. Chang
Barefoot Networks
October 30, 2017

VXLAN-GPE Encapsulation for In-situ OAM Data
draft-brockners-ioam-vxlan-gpe-00

Abstract

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. This document outlines how IOAM data fields are encapsulated in VXLAN-GPE.

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 May 3, 2018.

Copyright Notice

Copyright (c) 2017 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
2. Conventions	3
2.1. Requirement Language	3
2.2. Abbreviations	3
3. IOAM Data Field Encapsulation in VXLAN-GPE	3
3.1. IOAM Trace Data in VXLAN-GPE	3
3.2. IOAM POT Data in VXLAN-GPE	7
3.3. IOAM Edge-to-Edge Data in VXLAN-GPE	8
4. Discussion of the encapsulation approach	9
5. IANA Considerations	10
6. Security Considerations	10
7. Acknowledgements	11
8. References	11
8.1. Normative References	11
8.2. Informative References	12
Authors' Addresses	12

1. Introduction

In-situ OAM (IOAM) records OAM information within the packet while the packet traverses a particular network domain. The term "in-situ" refers to the fact that the IOAM data fields are added to the data packets rather than being sent within packets specifically dedicated to OAM. This document defines how IOAM data fields are transported as part of the VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe] encapsulation. The IOAM data fields are defined in [I-D.ietf-ippm-ioam-data]. An implementation of IOAM which leverages VXLAN-GPE to carry the IOAM data is available from the FD.io open source software project [FD.io].

2. Conventions

2.1. Requirement Language

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

2.2. Abbreviations

Abbreviations used in this document:

IOAM: In-situ Operations, Administration, and Maintenance

MTU: Maximum Transmit Unit

OAM: Operations, Administration, and Maintenance

POT: Proof of Transit

SFC: Service Function Chain

VXLAN-GPE: Virtual eXtensible Local Area Network, Generic Protocol Extension

3. IOAM Data Field Encapsulation in VXLAN-GPE

For encapsulating IOAM data fields into VXLAN-GPE [I-D.ietf-nvo3-vxlan-gpe] the different IOAM data fields are added as options within new IOAM protocol headers in VXLAN-GPE. In an administrative domain where IOAM is used, insertion of the IOAM protocol header(s) in VXLAN GPE is enabled at the VXLAN-GPE tunnel endpoints which also serve as IOAM encapsulating/decapsulating nodes by means of configuration. The VXLAN-GPE header is defined in [I-D.ietf-nvo3-vxlan-gpe]. IOAM specific fields for VXLAN-GPE are defined in this document.

3.1. IOAM Trace Data in VXLAN-GPE

IOAM tracing data represents data that is inserted at nodes that a packet traverses. To allow for optimal implementations in both software as well as hardware forwarders, two different ways to encapsulate IOAM data are defined: "Pre-allocated" and "Incremental". See [I-D.ietf-ippm-ioam-data] for details on IOAM tracing and the pre-allocated and incremental IOAM trace options.

The packet formats of the pre-allocated IOAM trace and incremental IOAM trace when encapsulated in VXLAN-GPE are defined as below.

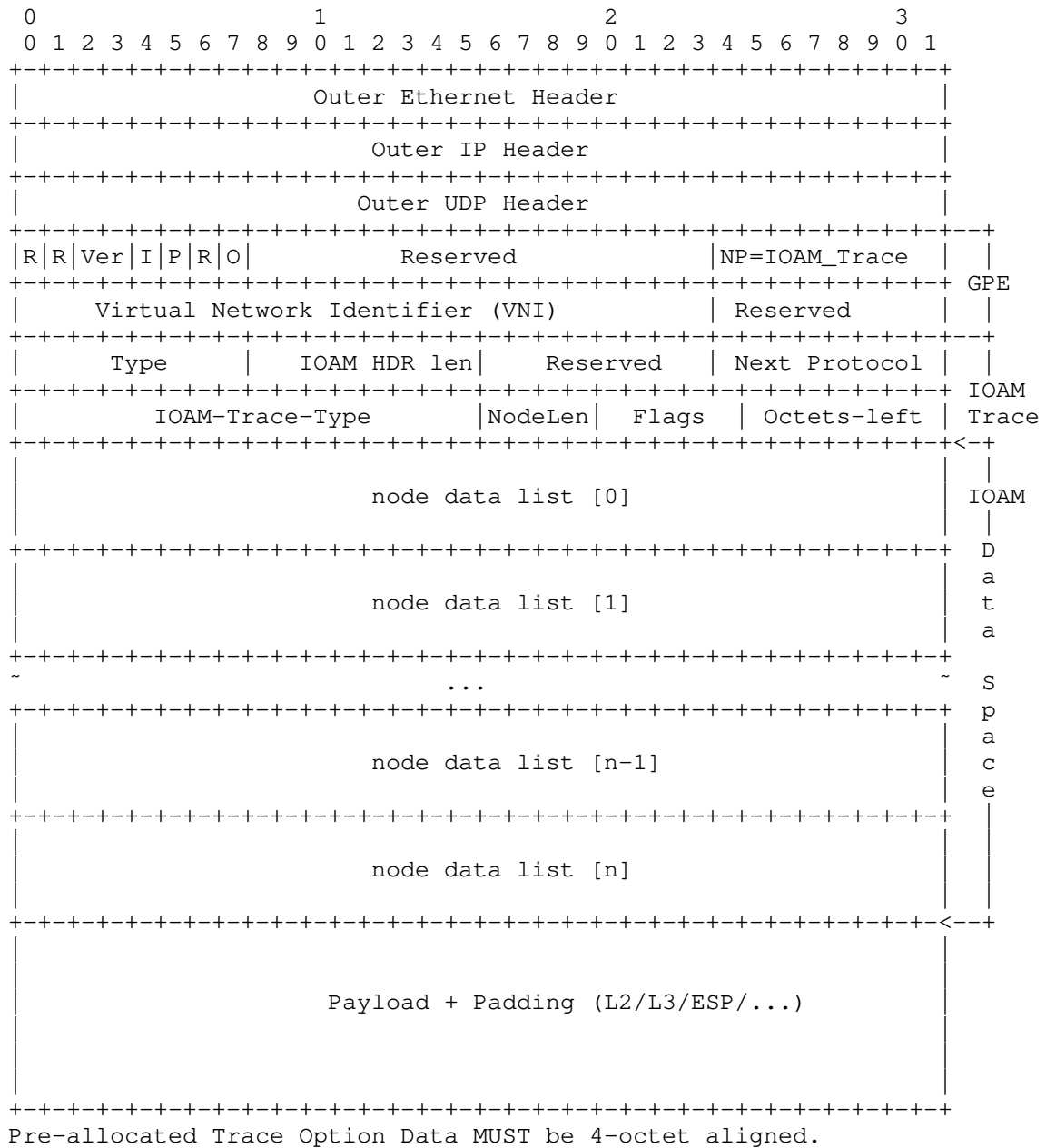


Figure 1: IOAM Pre-allocated Trace Option Format over VXLAN-GPE

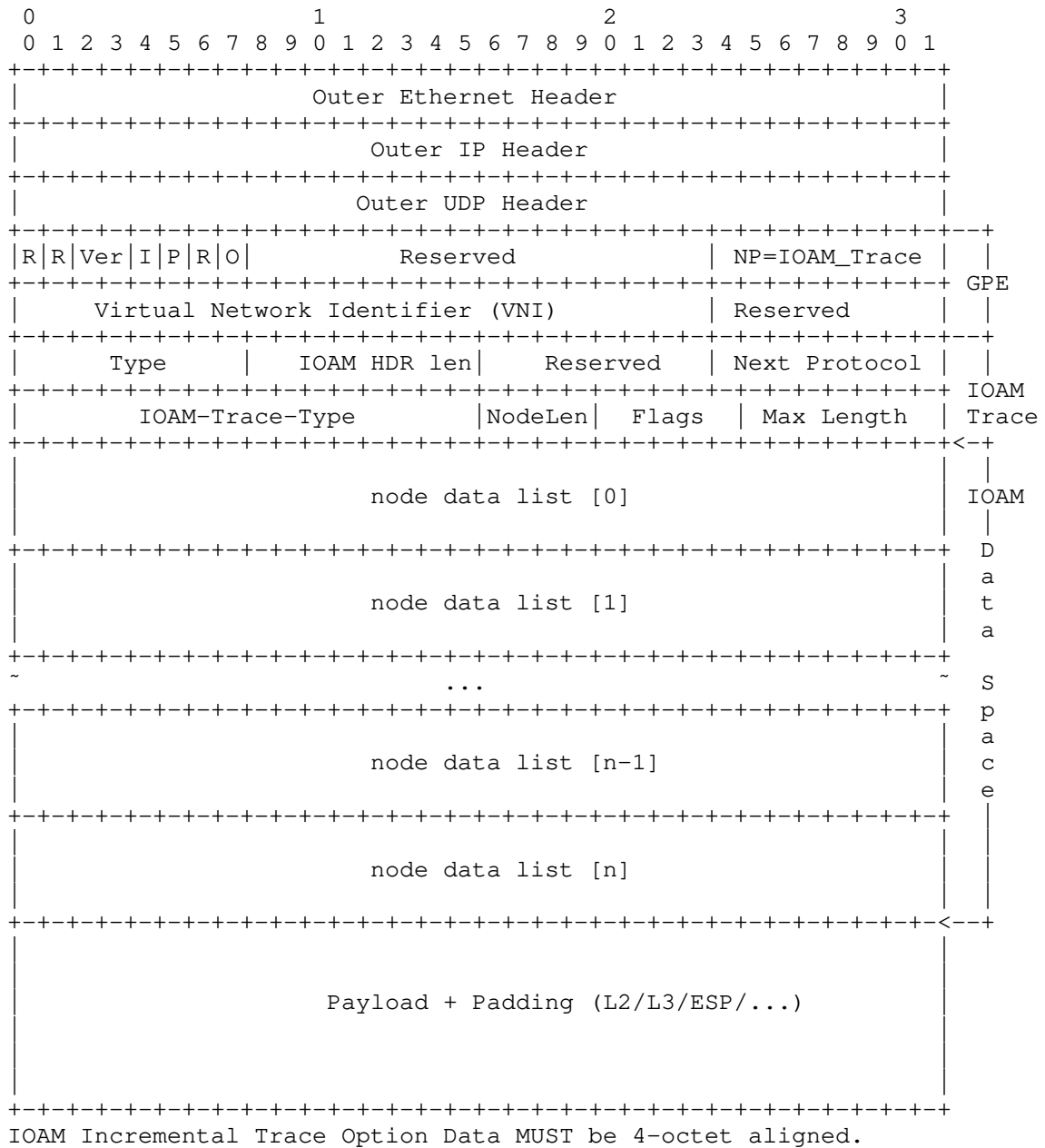


Figure 2: IOAM Incremental Trace Option Format over VXLAN-GPE

The IOAM Trace header consists of 8 octets, as illustrated in Figure 1 and Figure 2. The format of the first 4 octets (Figure 3)

is specific to VXLAN-GPE, and is defined in this document. The format of the next 4 octets (trace option header) is defined in [I-D.ietf-ippm-ioam-data], and is described here for the sake of clarity.

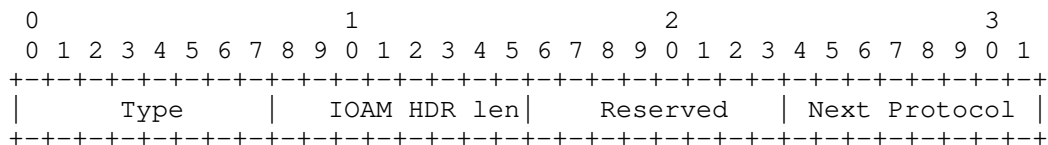


Figure 3: Trace Shim Header for VXLAN-GPE

The fields of the trace shim header are as follows:

Type: 8-bit unsigned integer defining IOAM header type
IOAM_TRACE_Preallocated or IOAM_Trace_Incremental are defined
here.

IOAM HDR len: 8-bit unsigned integer. Length of the IOAM HDR in 4-octet units.

Reserved: 8-bit reserved field MUST be set to zero.

Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol. The value is from the IANA registry setup for VXLAN GPE Next Protocol defined in [I-D.ietf-nvo3-vxlan-gpe].

The fields of the trace option header [I-D.ietf-ippm-ioam-data] are as follows:

IOAM-Trace-Type: 16-bit identifier of IOAM Trace Type as defined in [I-D.ietf-ippm-ioam-data] IOAM-Trace-Types.

Node Data Length: 4-bit unsigned integer as defined in [I-D.ietf-ippm-ioam-data].

Flags: 5-bit field as defined in [I-D.ietf-ippm-ioam-data].

Octets-left: 7-bit unsigned integer as defined in [I-D.ietf-ippm-ioam-data].

Maximum-length: 7-bit unsigned integer as defined in [I-D.ietf-ippm-ioam-data].

Node data List [n]: Variable-length field as defined in [I-D.ietf-ippm-ioam-data].

3.2. IOAM POT Data in VXLAN-GPE

IOAM proof of transit (POT, see also [I-D.brockners-proof-of-transit]) offers a means to verify that a packet has traversed a defined set of nodes. IOAM POT data fields are encapsulated in VXLAN-GPE using a dedicated VXLAN-GPE protocol header:

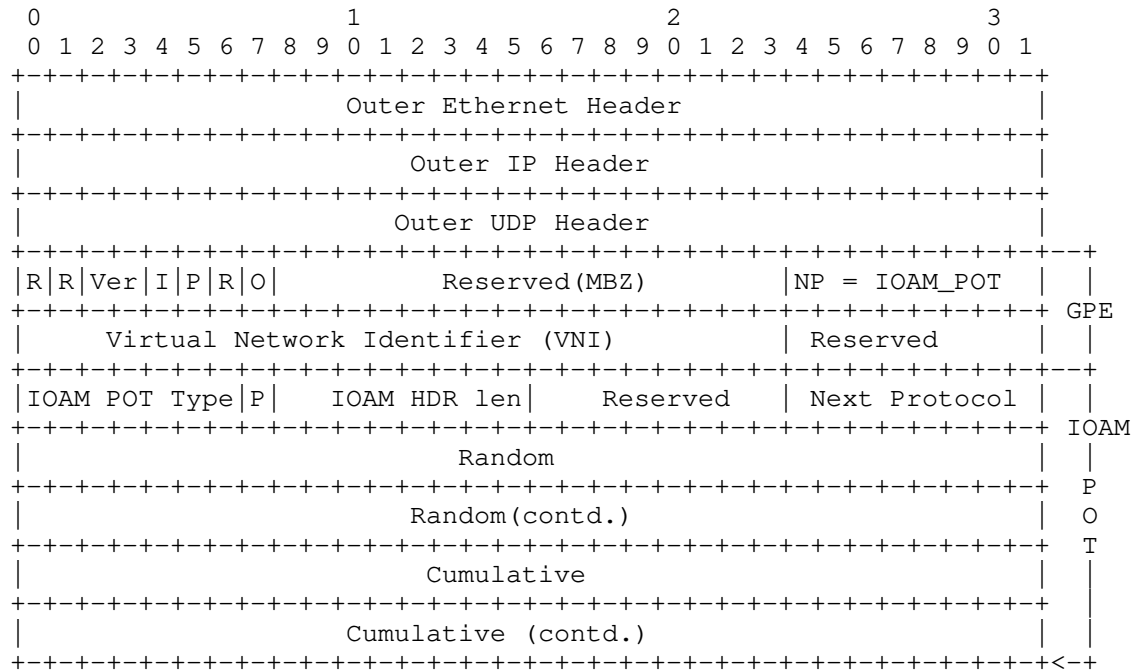


Figure 4: IOAM POT Header Following the VXLAN-GPE Header

The IOAM POT Shim Header (Figure 5), which is defined in this document, is a 4-octet header, that includes the following fields:

IOAM POT Type: 7-bit identifier of a particular POT variant that specifies the POT data that is to be included as defined in [I-D.ietf-ippm-ioam-data].

Profile to use (P): 1-bit as defined in [I-D.ietf-ippm-ioam-data] IOAM POT Option.

IOAM HDR len: 8-bit unsigned integer. Length of the IOAM HDR in 4-octet units.

Reserved: 8-bit reserved field MUST be set to zero.

Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol. The value is from the IANA registry setup for VXLAN GPE Next Protocol defined in [I-D.ietf-nvo3-vxlan-gpe].

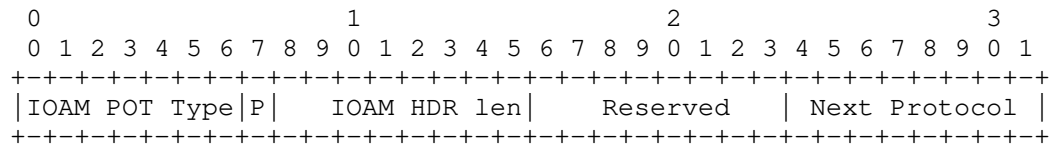


Figure 5: POT Shim Header for VXLAN-GPE

The rest of the fields in the POT option [I-D.ietf-ippm-ioam-data] are as follows:

Random: 64-bit Per-packet random number.

Cumulative: 64-bit Cumulative value that is updated by the Service Functions.

3.3. IOAM Edge-to-Edge Data in VXLAN-GPE

The IOAM edge-to-edge option is to carry data that is added by the IOAM encapsulating node and interpreted by the IOAM decapsulating node. IOAM specific fields to encapsulate IOAM Edge-to-Edge data fields are defined as follows:

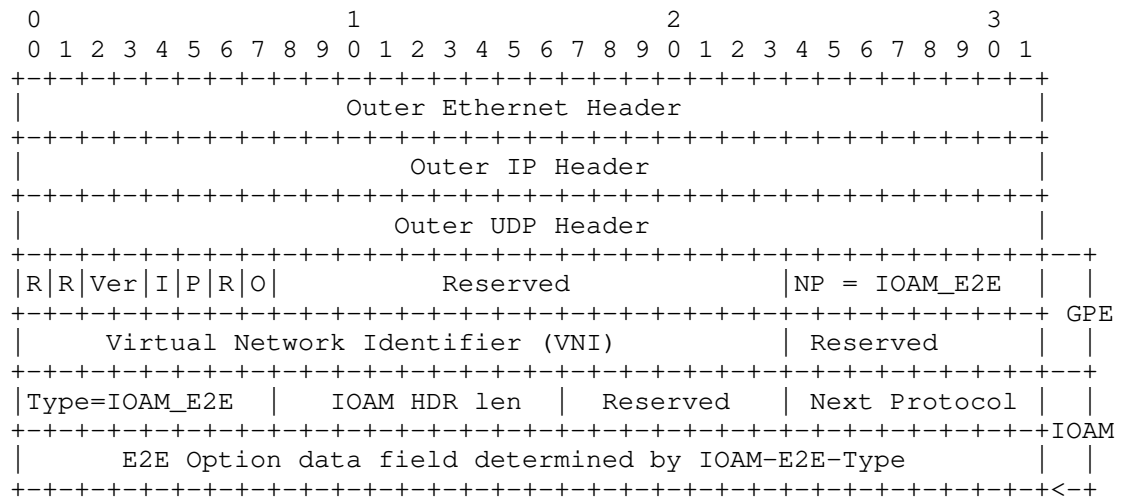


Figure 6: IOAM Edge-to-Edge over a VXLAN-GPE Header

The IOAM E2E Shim Header, which is defined in this document, is a 4-octet header, that includes the following fields:

Type: 8-bit identifier of a particular E2E variant that specifies the E2E data that is to be included as defined in [I-D.ietf-ippm-ioam-data].

IOAM HDR len: 8-bit unsigned integer. Length of the IOAM HDR in 4-octet units.

Reserved: 8-bit reserved field MUST be set to zero.

Next Protocol: 8-bit unsigned integer that determines the type of header following IOAM protocol. The value is from the IANA registry setup for VXLAN GPE Next Protocol defined in [I-D.ietf-nvo3-vxlan-gpe].

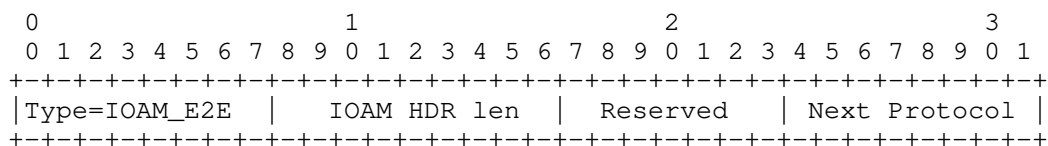


Figure 7: E2E Shim Header for VXLAN-GPE

The rest of the E2E option [I-D.ietf-ippm-ioam-data] consists of:

E2E Option data field: Variable length field as defined in [I-D.ietf-ippm-ioam-data] IOAM E2E Option.

4. Discussion of the encapsulation approach

This section is to support the working group discussion in selecting the most appropriate approach for encapsulating IOAM data fields in VXLAN-GPE.

An encapsulation of IOAM data fields in VXLAN-GPE should be friendly to an implementation in both hardware as well as software forwarders. Hardware forwarders benefit from an encapsulation that minimizes iterative look-ups of fields within the packet: Any operation which looks up the value of a field within the packet, based on which another lookup is performed, consumes additional gates and time in an implementation - both of which are desired to be kept to a minimum. This means that flat TLV structures are to be preferred over nested TLV structures. IOAM data fields are grouped into three option categories: Trace, proof-of-transit, and edge-to-edge. Each of these three options defines a TLV structure. A hardware-friendly encapsulation approach avoids grouping these three option categories

into yet another TLV structure, but would rather carry the options as a serial sequence.

Two approaches for encapsulating IOAM data fields in VXLAN-GPE could be considered:

1. Use a single GPE protocol type for all IOAM types: IOAM would receive a single GPE protocol type code point. A "sub-type" (e.g. 4 bit wide) would then specify what IOAM options type(s) (trace, proof-of-transit, edge-to-edge) are carried. In case there is a need for additional IOAM options, changes would be contained within the single GPE protocol type for IOAM.
2. Use one GPE protocol type per IOAM options type: Each IOAM data field option (trace, proof-of-transit, and edge-to-edge) would be specified by its own "next protocol", i.e. each IOAM options type becomes its own GPE protocol type with a dedicated code point. This implies that in case additional IOAM option types would be added in the future, additional GPE protocol type code points would need to be allocated.

The second option has been chosen here, because it avoids the additional layer of TLV nesting that the use of a single GPE protocol type for all IOAM option types would result in.

5. IANA Considerations

IANA is requested to allocate protocol numbers for the following VXLAN-GPE "Next Protocols" related to IOAM:

Next Protocol	Description	Reference
x	IOAM_Trace	This document
y	IOAM_POT	This document
z	IOAM_E2E	This document

6. Security Considerations

The security considerations of VXLAN-GPE are discussed in [I-D.ietf-nvo3-vxlan-gpe], and the security considerations of IOAM in general are discussed in [I-D.ietf-ipm-ioam-data].

IOAM is considered a "per domain" feature, where one or several operators decide on leveraging and configuring IOAM according to their needs. Still, operators need to properly secure the IOAM

domain to avoid malicious configuration and use, which could include injecting malicious IOAM packets into a domain.

7. Acknowledgements

The authors would like to thank Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, Stefano Previdi, Hemant Singh, Erik Nordmark, LJ Wobker, and Andrew Yourtchenko for the comments and advice.

8. References

8.1. Normative References

- [ETYPES] "IANA Ethernet Numbers",
<<https://www.iana.org/assignments/ethernet-numbers/ethernet-numbers.xhtml>>.
- [I-D.ietf-ippm-ioam-data]
Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., Chang, R., and d. daniel.bernier@bell.ca, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-00 (work in progress), September 2017.
- [I-D.ietf-nvo3-vxlan-gpe]
Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN", draft-ietf-nvo3-vxlan-gpe-04 (work in progress), April 2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <<https://www.rfc-editor.org/info/rfc2784>>.
- [RFC3232] Reynolds, J., Ed., "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, DOI 10.17487/RFC3232, January 2002, <<https://www.rfc-editor.org/info/rfc3232>>.

8.2. Informative References

- [FD.io] "Fast Data Project: FD.io", <<https://fd.io/>>.
- [I-D.brockners-proof-of-transit]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C.,
Leddy, J., Youell, S., Mozes, D., and T. Mizrahi, "Proof
of Transit", draft-brockners-proof-of-transit-03 (work in
progress), March 2017.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function
Chaining (SFC) Architecture", RFC 7665,
DOI 10.17487/RFC7665, October 2015, <[https://www.rfc-
editor.org/info/rfc7665](https://www.rfc-editor.org/info/rfc7665)>.

Authors' Addresses

Frank Brockners
Cisco Systems, Inc.
Hansaallee 249, 3rd Floor
DUESSELDORF, NORDRHEIN-WESTFALEN 40549
Germany

Email: fbrockne@cisco.com

Shwetha Bhandari
Cisco Systems, Inc.
Cessna Business Park, Sarjapura Marathalli Outer Ring Road
Bangalore, KARNATAKA 560 087
India

Email: shwethab@cisco.com

Vengada Prasad Govindan
Cisco Systems, Inc.

Email: venggovi@cisco.com

Carlos Pignataro
Cisco Systems, Inc.
7200-11 Kit Creek Road
Research Triangle Park, NC 27709
United States

Email: cpignata@cisco.com

Hannes Gredler
RtBrick Inc.

Email: hannes@rtbrick.com

John Leddy
Comcast

Email: John_Leddy@cable.comcast.com

Stephen Youell
JP Morgan Chase
25 Bank Street
London E14 5JP
United Kingdom

Email: stephen.youell@jpmorgan.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam 20692
Israel

Email: talmi@marvell.com

David Mozes
Mellanox Technologies Ltd.

Email: davidm@mellanox.com

Petr Lapukhov
Facebook
1 Hacker Way
Menlo Park, CA 94025
US

Email: petr@fb.com

Remy Chang
Barefoot Networks
2185 Park Boulevard
Palo Alto, CA 94306
US

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 28, 2018

F. Maino
V. Ermagan
Cisco Systems
A. Cabellos
Universitat Politecnica de Catalunya
D. Saucez
INRIA
October 25, 2017

LISP-Security (LISP-SEC)
draft-ietf-lisp-sec-14

Abstract

This memo specifies LISP-SEC, a set of security mechanisms that provides origin authentication, integrity and anti-replay protection to LISP's EID-to-RLOC mapping data conveyed via mapping lookup process. LISP-SEC also enables verification of authorization on EID-prefix claims in Map-Reply messages.

Requirements Language

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

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <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 April 28, 2018.

Copyright Notice

Copyright (c) 2017 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	3
3. LISP-SEC Threat Model	4
4. Protocol Operations	5
5. LISP-SEC Control Messages Details	7
5.1. Encapsulated Control Message LISP-SEC Extensions	7
5.2. Map-Reply LISP-SEC Extensions	9
5.3. Map-Register LISP-SEC Extensions	11
5.4. ITR Processing	11
5.4.1. Map-Reply Record Validation	13
5.4.2. PITR Processing	14
5.5. Encrypting and Decrypting an OTK	14
5.6. Map-Resolver Processing	15
5.7. Map-Server Processing	15
5.7.1. Map-Server Processing in Proxy mode	16
5.8. ETR Processing	16
6. Security Considerations	17
6.1. Mapping System Security	17
6.2. Random Number Generation	17
6.3. Map-Server and ETR Colocation	17
6.4. Deploying LISP-SEC	17
6.5. Shared Keys Provisioning	18
6.6. Replay Attacks	18
6.7. Denial of Service and Distributed Denial of Service Attacks	19
7. IANA Considerations	19
7.1. ECM AD Type Registry	19
7.2. Map-Reply AD Type Registry	19
7.3. HMAC Functions	20
7.4. Key Wrap Functions	20
7.5. Key Derivation Functions	21

8. Acknowledgements	21
9. Normative References	21
Authors' Addresses	22

1. Introduction

The Locator/ID Separation Protocol [RFC6830] is a network-layer-based protocol that enables separation of IP addresses into two new numbering spaces: Endpoint Identifiers (EIDs) and Routing Locators (RLOCs). EID-to-RLOC mappings are stored in a database, the LISP Mapping System, and made available via the Map-Request/Map-Reply lookup process. If these EID-to-RLOC mappings, carried through Map-Reply messages, are transmitted without integrity protection, an adversary can manipulate them and hijack the communication, impersonate the requested EID, or mount Denial of Service or Distributed Denial of Service attacks. Also, if the Map-Reply message is transported unauthenticated, an adversarial LISP entity can overclaim an EID-prefix and maliciously redirect traffic directed to a large number of hosts. The LISP-SEC threat model, described in Section 3, is built on top of the LISP threat model defined in [RFC7835], that includes a detailed description of "overclaiming" attack.

This memo specifies LISP-SEC, a set of security mechanisms that provides origin authentication, integrity and anti-replay protection to LISP's EID-to-RLOC mapping data conveyed via mapping lookup process. LISP-SEC also enables verification of authorization on EID-prefix claims in Map-Reply messages, ensuring that the sender of a Map-Reply that provides the location for a given EID-prefix is entitled to do so according to the EID prefix registered in the associated Map-Server. Map-Register security, including the right for a LISP entity to register an EID-prefix or to claim presence at an RLOC, is out of the scope of LISP-SEC. Additional security considerations are described in Section 6.

2. Definition of Terms

One-Time Key (OTK): An ephemeral randomly generated key that must be used for a single Map-Request/Map-Reply exchange.

ITR One-Time Key (ITR-OTK): The One-Time Key generated at the ITR.

MS One-Time Key (MS-OTK): The One-Time Key generated at the Map-Server.

Authentication Data (AD): Metadata that is included either in a LISP Encapsulated Control Message (ECM) header, as defined in Section 6.1.8 of [RFC6830], or in a Map-Reply message to support

confidentiality, integrity protection, and verification of EID-prefix authorization.

OTK Authentication Data (OTK-AD): The portion of ECM Authentication Data that contains a One-Time Key.

EID Authentication Data (EID-AD): The portion of ECM and Map-Reply Authentication Data used for verification of EID-prefix authorization.

Packet Authentication Data (PKT-AD): The portion of Map-Reply Authentication Data used to protect the integrity of the Map-Reply message.

For definitions of other terms, notably Map-Request, Map-Reply, Ingress Tunnel Router (ITR), Egress Tunnel Router (ETR), Map-Server (MS), and Map-Resolver (MR) please consult the LISP specification [RFC6830].

3. LISP-SEC Threat Model

LISP-SEC addresses the control plane threats, described in section 3.7 and 3.8 of [RFC7835], that target EID-to-RLLOC mappings, including manipulations of Map-Request and Map-Reply messages, and malicious ETR EID prefix overclaiming. LISP-SEC makes two main assumptions: (1) the LISP mapping system is expected to deliver a Map-Request message to their intended destination ETR as identified by the EID, and (2) no man-in-the-middle (MITM) attack can be mounted within the LISP Mapping System. How the Mapping System is protected from MITM attacks depends from the particular Mapping Systems used, and is out of the scope of this memo. Furthermore, while LISP-SEC enables detection of EID prefix overclaiming attacks, it assumes that Map-Servers can verify the EID prefix authorization at time of registration.

According to the threat model described in [RFC7835] LISP-SEC assumes that any kind of attack, including MITM attacks, can be mounted in the access network, outside of the boundaries of the LISP mapping system. An on-path attacker, outside of the LISP mapping system can, for example, hijack Map-Request and Map-Reply messages, spoofing the identity of a LISP node. Another example of on-path attack, called overclaiming attack, can be mounted by a malicious Egress Tunnel Router (ETR), by overclaiming the EID-prefixes for which it is authoritative. In this way the ETR can maliciously redirect traffic directed to a large number of hosts.

4. Protocol Operations

The goal of the security mechanisms defined in [RFC6830] is to prevent unauthorized insertion of mapping data by providing origin authentication and integrity protection for the Map-Registration, and by using the nonce to detect unsolicited Map-Reply sent by off-path attackers.

LISP-SEC builds on top of the security mechanisms defined in [RFC6830] to address the threats described in Section 3 by leveraging the trust relationships existing among the LISP entities participating to the exchange of the Map-Request/Map-Reply messages. Those trust relationships are used to securely distribute a One-Time Key (OTK) that provides origin authentication, integrity and anti-replay protection to mapping data conveyed via the mapping lookup process, and that effectively prevent overclaiming attacks. The processing of security parameters during the Map-Request/Map-Reply exchange is as follows:

- o The ITR-OTK is generated and stored at the ITR, and securely transported to the Map-Server.
- o The Map-Server uses the ITR-OTK to compute a Keyed-Hashing for Message Authentication (HMAC) [RFC2104] that protects the integrity of the mapping data known to the Map-Server to prevent overclaiming attacks. The Map-Server also derives a new OTK, the MS-OTK, that is passed to the ETR, by applying a Key Derivation Function (KDF) to the ITR-OTK.
- o The ETR uses the MS-OTK to compute an HMAC that protects the integrity of the Map-Reply sent to the ITR.
- o Finally, the ITR uses the stored ITR-OTK to verify the integrity of the mapping data provided by both the Map-Server and the ETR, and to verify that no overclaiming attacks were mounted along the path between the Map-Server and the ITR.

Section 5 provides the detailed description of the LISP-SEC control messages and their processing, while the rest of this section describes the flow of protocol operations at each entity involved in the Map-Request/Map-Reply exchange:

- o The ITR, upon needing to transmit a Map-Request message, generates and stores an OTK (ITR-OTK). This ITR-OTK is included into the Encapsulated Control Message (ECM) that contains the Map-Request sent to the Map-Resolver. To provide confidentiality to the ITR-OTK over the path between the ITR and its Map-Resolver, the ITR-OTK SHOULD be encrypted using a preconfigured key shared between

the ITR and the Map-Resolver, similar to the key shared between the ETR and the Map-Server in order to secure ETR registration [RFC6833].

- o The Map-Resolver decapsulates the ECM message, decrypts the ITR-OTK, if needed, and forwards through the Mapping System the received Map-Request and the ITR-OTK, as part of a new ECM message. As described in Section 5.6, the LISP Mapping System delivers the ECM to the appropriate Map-Server, as identified by the EID destination address of the Map-Request.
- o The Map-Server is configured with the location mappings and policy information for the ETR responsible for the EID destination address. Using this preconfigured information, the Map-Server, after the decapsulation of the ECM message, finds the longest match EID-prefix that covers the requested EID in the received Map-Request. The Map-Server adds this EID-prefix, together with an HMAC computed using the ITR-OTK, to a new Encapsulated Control Message that contains the received Map-Request.
- o The Map-Server derives a new OTK, the MS-OTK, by applying a Key Derivation Function (KDF) to the ITR-OTK. This MS-OTK is included in the Encapsulated Control Message that the Map-Server uses to forward the Map-Request to the ETR. To provide MS-OTK confidentiality over the path between the Map-Server and the ETR, the MS-OTK SHOULD be encrypted using the key shared between the ETR and the Map-Server in order to secure ETR registration [RFC6833].
- o If the Map-Server is acting in proxy mode, as specified in [RFC6830], the ETR is not involved in the generation of the Map-Reply. In this case the Map-Server generates the Map-Reply on behalf of the ETR as described below.
- o The ETR, upon receiving the ECM encapsulated Map-Request from the Map-Server, decrypts the MS-OTK, if needed, and originates a standard Map-Reply that contains the EID-to-RLLOC mapping information as specified in [RFC6830].
- o The ETR computes an HMAC over this standard Map-Reply, keyed with MS-OTK to protect the integrity of the whole Map-Reply. The ETR also copies the EID-prefix authorization data that the Map-Server included in the ECM encapsulated Map-Request into the Map-Reply message. The ETR then sends this complete Map-Reply message to the requesting ITR.
- o The ITR, upon receiving the Map-Reply, uses the locally stored ITR-OTK to verify the integrity of the EID-prefix authorization

data included in the Map-Reply by the Map-Server. The ITR computes the MS-OTK by applying the same KDF used by the Map-Server, and verifies the integrity of the Map-Reply. If the integrity checks fail, the Map-Reply MUST be discarded. Also, if the EID-prefixes claimed by the ETR in the Map-Reply are not equal or more specific than the EID-prefix authorization data inserted by the Map-Server, the ITR MUST discard the Map-Reply.

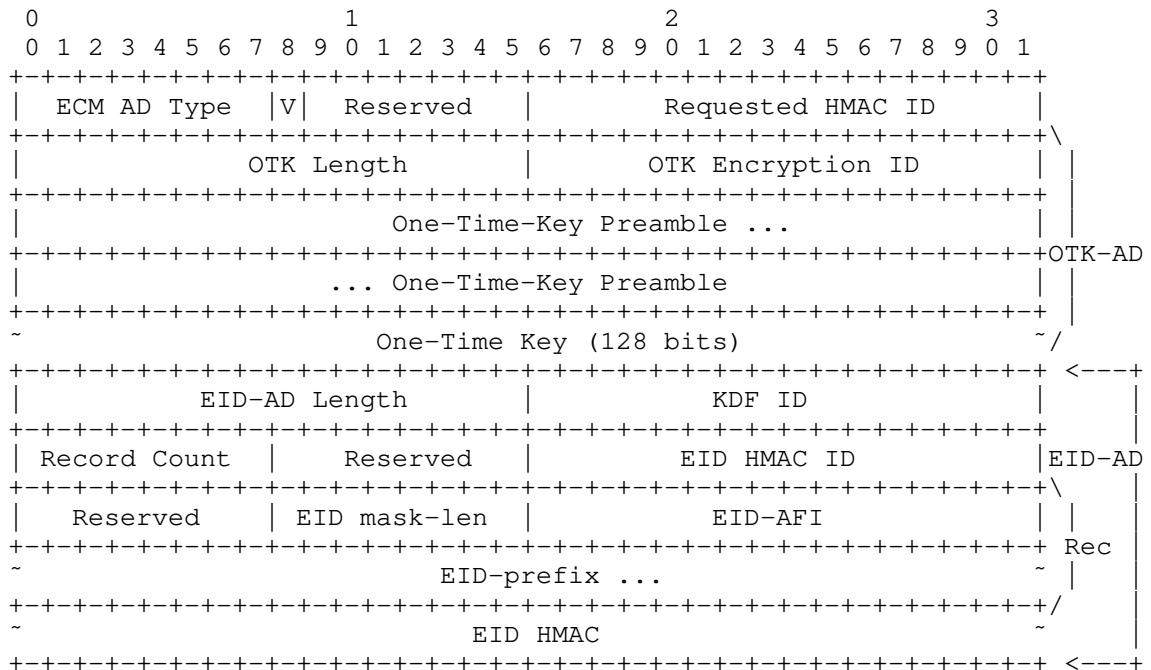
5. LISP-SEC Control Messages Details

LISP-SEC metadata associated with a Map-Request is transported within the Encapsulated Control Message that contains the Map-Request.

LISP-SEC metadata associated with the Map-Reply is transported within the Map-Reply itself.

5.1. Encapsulated Control Message LISP-SEC Extensions

LISP-SEC uses the ECM (Encapsulated Control Message) defined in [RFC6830] with Type set to 8, and S bit set to 1 to indicate that the LISP header includes Authentication Data (AD). The format of the LISP-SEC ECM Authentication Data is defined in the following figure. OTK-AD stands for One-Time Key Authentication Data and EID-AD stands for EID Authentication Data.



LISP-SEC ECM Authentication Data

ECM AD Type: 1 (LISP-SEC Authentication Data). See Section 7.

V: Key Version bit. This bit is toggled when the sender switches to a new OTK wrapping key

Reserved: Set to 0 on transmission and ignored on receipt.

Requested HMAC ID: The HMAC algorithm requested by the ITR. See Section 5.4 for details.

OTK Length: The length (in bytes) of the OTK Authentication Data (OTK-AD), that contains the OTK Preamble and the OTK.

OTK Encryption ID: The identifier of the key wrapping algorithm used to encrypt the One-Time-Key. When a 128-bit OTK is sent unencrypted by the Map-Resolver, the OTK Encryption ID is set to NULL_KEY_WRAP_128. See Section 5.5 for more details.

One-Time-Key Preamble: set to 0 if the OTK is not encrypted. When the OTK is encrypted, this field MAY carry additional metadata resulting from the key wrapping operation. When a 128-bit OTK is

sent unencrypted by Map-Resolver, the OTK Preamble is set to 0x0000000000000000 (64 bits). See Section 5.5 for details.

One-Time-Key: the OTK encrypted (or not) as specified by OTK Encryption ID. See Section 5.5 for details.

EID-AD Length: length (in bytes) of the EID Authentication Data (EID-AD). The ITR MUST set EID-AD Length to 4 bytes, as it only fills the KDF ID field, and all the remaining fields part of the EID-AD are not present. An EID-AD MAY contain multiple EID-records. Each EID-record is 4-byte long plus the length of the AFI-encoded EID-prefix.

KDF ID: Identifier of the Key Derivation Function used to derive the MS-OTK. The ITR MAY use this field to indicate the recommended KDF algorithm, according to local policy. The Map-Server can overwrite the KDF ID if it does not support the KDF ID recommended by the ITR. See Section 5.4 for more details.

Record Count: The number of records in this Map-Request message. A record is comprised of the portion of the packet that is labeled 'Rec' above and occurs the number of times equal to Record Count.

Reserved: Set to 0 on transmission and ignored on receipt.

EID HMAC ID: Identifier of the HMAC algorithm used to protect the integrity of the EID-AD. This field is filled by Map-Server that computed the EID-prefix HMAC. See Section 5.4 for more details.

EID mask-len: Mask length for EID-prefix.

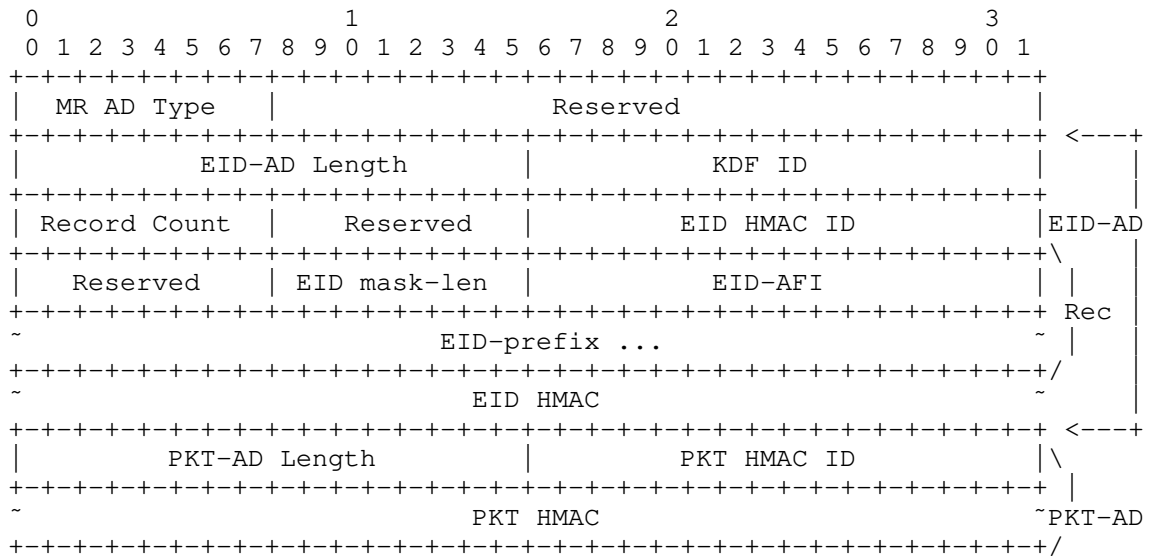
EID-AFI: Address family of EID-prefix according to [RFC5226]

EID-prefix: The Map-Server uses this field to specify the EID-prefix that the destination ETR is authoritative for, and is the longest match for the requested EID.

EID HMAC: HMAC of the EID-AD computed and inserted by Map-Server. Before computing the HMAC operation the EID HMAC field MUST be set to 0. The HMAC covers the entire EID-AD.

5.2. Map-Reply LISP-SEC Extensions

LISP-SEC uses the Map-Reply defined in [RFC6830], with Type set to 2, and S bit set to 1 to indicate that the Map-Reply message includes Authentication Data (AD). The format of the LISP-SEC Map-Reply Authentication Data is defined in the following figure. PKT-AD is the Packet Authentication Data that covers the Map-Reply payload.



LISP-SEC Map-Reply Authentication Data

MR AD Type: 1 (LISP-SEC Authentication Data). See Section 7.

EID-AD Length: length (in bytes) of the EID-AD. An EID-AD MAY contain multiple EID-records. Each EID-record is 4-byte long plus the length of the AFI-encoded EID-prefix.

KDF ID: Identifier of the Key Derivation Function used to derive MS-OTK. See Section 5.7 for more details.

Record Count: The number of records in this Map-Reply message. A record is comprised of the portion of the packet that is labeled 'Rec' above and occurs the number of times equal to Record Count.

Reserved: Set to 0 on transmission and ignored on receipt.

EID HMAC ID: Identifier of the HMAC algorithm used to protect the integrity of the EID-AD. See Section 5.7 for more details.

EID mask-len: Mask length for EID-prefix.

EID-AFI: Address family of EID-prefix according to [RFC5226].

EID-prefix: This field contains an EID-prefix that the destination ETR is authoritative for, and is the longest match for the requested EID.

EID HMAC: HMAC of the EID-AD, as computed by the Map-Server. Before computing the HMAC operation the EID HMAC field MUST be set to 0. The HMAC covers the entire EID-AD.

PKT-AD Length: length (in bytes) of the Packet Authentication Data (PKT-AD).

PKT HMAC ID: Identifier of the HMAC algorithm used to protect the integrity of the Map-reply.

PKT HMAC: HMAC of the whole Map-Reply packet, including the LISP-SEC Authentication Data. The scope of the authentication goes from the Map-Reply Type field to the PKT HMAC field included. Before computing the HMAC operation the PKT HMAC field MUST be set to 0. See Section 5.8 for more details.

5.3. Map-Register LISP-SEC Extensions

This memo is allocating one of the bits marked as Reserved in the Map-Register message defined in Section 6.1.6 of [RFC6830]. More precisely, the second bit after the Type field in a Map-Register message is allocated as the S bit. The S bit indicates to the Map-Server that the registering ETR is LISP-SEC enabled. An ETR that supports LISP-SEC MUST set the S bit in its Map-Register messages.

5.4. ITR Processing

Upon creating a Map-Request, the ITR generates a random ITR-OTK that is stored locally, together with the nonce generated as specified in [RFC6830].

The Map-Request MUST be encapsulated in an ECM, with the S-bit set to 1, to indicate the presence of Authentication Data. If the ITR and the Map-Resolver are configured with a shared key, the ITR-OTK confidentiality SHOULD be protected by wrapping the ITR-OTK with the algorithm specified by the OTK Encryption ID field. See Section 5.5 for further details on OTK encryption.

The Requested HMAC ID field contains the suggested HMAC algorithm to be used by the Map-Server and the ETR to protect the integrity of the ECM Authentication data and of the Map-Reply.

The KDF ID field specifies the suggested key derivation function to be used by the Map-Server to derive the MS-OTK. A KDF Value of NONE (0), MAY be used to specify that the ITR has no preferred KDF ID.

The EID-AD length is set to 4 bytes, since the Authentication Data does not contain EID-prefix Authentication Data, and the EID-AD contains only the KDF ID field.

In response to an encapsulated Map-Request that has the S-bit set, an ITR MUST receive a Map-Reply with the S-bit set, that includes an EID-AD and a PKT-AD. If the Map-Reply does not include both ADs, the ITR MUST discard it. In response to an encapsulated Map-Request with S-bit set to 0, the ITR expects a Map-Reply with S-bit set to 0, and the ITR SHOULD discard the Map-Reply if the S-bit is set.

Upon receiving a Map-Reply, the ITR must verify the integrity of both the EID-AD and the PKT-AD, and MUST discard the Map-Reply if one of the integrity checks fails. After processing the Map-Reply, the ITR must discard the <nonce, ITR-OTK> pair associated to the Map-Reply

The integrity of the EID-AD is verified using the locally stored ITR-OTK to re-compute the HMAC of the EID-AD using the algorithm specified in the EID HMAC ID field. If the EID HMAC ID field does not match the Requested HMAC ID the ITR SHOULD discard the Map-Reply and send, at the first opportunity it needs to, a new Map-Request with a different Requested HMAC ID field, according to ITR's local policy. The scope of the HMAC operation covers the entire EID-AD, from the EID-AD Length field to the EID HMAC field, which must be set to 0 before the computation of the HMAC.

ITR MUST set the EID HMAC ID field to 0 before computing the HMAC.

To verify the integrity of the PKT-AD, first the MS-OTK is derived from the locally stored ITR-OTK using the algorithm specified in the KDF ID field. This is because the PKT-AD is generated by the ETR using the MS-OTK. If the KDF ID in the Map-Reply does not match the KDF ID requested in the Map-Request, the ITR SHOULD discard the Map-Reply and send, at the first opportunity it needs to, a new Map-Request with a different KDF ID, according to ITR's local policy.

The derived MS-OTK is then used to re-compute the HMAC of the PKT-AD using the Algorithm specified in the PKT HMAC ID field. If the PKT HMAC ID field does not match the Requested HMAC ID the ITR SHOULD discard the Map-Reply and send, at the first opportunity it needs to, a new Map-Request with a different Requested HMAC ID according to ITR's local policy or until all HMAC IDs supported by the ITR have been attempted.

Each individual Map-Reply EID-record is considered valid only if: (1) both EID-AD and PKT-AD are valid, and (2) the intersection of the EID-prefix in the Map-Reply EID-record with one of the EID-prefixes contained in the EID-AD is not empty. After identifying the Map-

Reply record as valid, the ITR sets the EID-prefix in the Map-Reply record to the value of the intersection set computed before, and adds the Map-Reply EID-record to its EID-to-RLOC cache, as described in [RFC6830]. An example of Map-Reply record validation is provided in Section 5.4.1.

The ITR SHOULD send SMR triggered Map-Requests over the mapping system in order to receive a secure Map-Reply. If an ITR accepts piggybacked Map-Replies, it SHOULD also send a Map-Request over the mapping system in order to verify the piggybacked Map-Reply with a secure Map-Reply.

5.4.1. Map-Reply Record Validation

The payload of a Map-Reply may contain multiple EID-records. The whole Map-Reply is signed by the ETR, with the PKT HMAC, to provide integrity protection and origin authentication to the EID-prefix records claimed by the ETR. The Authentication Data field of a Map-Reply may contain multiple EID-records in the EID-AD. The EID-AD is signed by the Map-Server, with the EID HMAC, to provide integrity protection and origin authentication to the EID-prefix records inserted by the Map-Server.

Upon receiving a Map-Reply with the S-bit set, the ITR first checks the validity of both the EID HMAC and of the PKT-AD HMAC. If either one of the HMACs is not valid, a log action MUST be taken and the Map-Reply MUST NOT be processed any further. If both HMACs are valid, the ITR proceeds with validating each individual EID-record claimed by the ETR by computing the intersection of each one of the EID-prefix contained in the payload of the Map-Reply with each one of the EID-prefixes contained in the EID-AD. An EID-record is valid only if at least one of the intersections is not the empty set.

For instance, the Map-Reply payload contains 3 mapping record EID-prefixes:

2001:db8:0102::/48

2001:db8:0103::/48

2001:db8:0200::/40

The EID-AD contains two EID-prefixes:

2001:db8:0103::/48

2001:db8:0203::/48

The EID-record with EID-prefix 2001:db8:0102::/48 is not eligible to be used by the ITR since it is not included in any of the EID-ADs signed by the Map-Server. A log action **MUST** be taken.

The EID-record with EID-prefix 2001:db8:0103::/48 is eligible to be used by the ITR because it matches the second EID-prefix contained in the EID-AD.

The EID-record with EID-prefix 2001:db8:0200::/40 is not eligible to be used by the ITR since it is not included in any of the EID-ADs signed by the Map-Server. A log action **MUST** be taken. In this last example the ETR is trying to over claim the EID-prefix 2001:db8:0200::/40, but the Map-Server authorized only 2001:db8:0203::/48, hence the EID-record is discarded.

5.4.2. PITR Processing

The processing performed by a PITR is equivalent to the processing of an ITR. However, if the PITR is directly connected to a Mapping System such as LISP+ALT [RFC6836], the PITR performs the functions of both the ITR and the Map-Resolver forwarding the Map-Request encapsulated in an ECM header that includes the Authentication Data fields as described in Section 5.6.

5.5. Encrypting and Decrypting an OTK

MS-OTK confidentiality is required in the path between the Map-Server and the ETR, the MS-OTK **SHOULD** be encrypted using the preconfigured key shared between the Map-Server and the ETR for the purpose of securing ETR registration [RFC6833]. Similarly, if ITR-OTK confidentiality is required in the path between the ITR and the Map-Resolver, the ITR-OTK **SHOULD** be encrypted with a key shared between the ITR and the Map-Resolver.

The OTK is encrypted using the algorithm specified in the OTK Encryption ID field. When the AES Key Wrap algorithm is used to encrypt a 128-bit OTK, according to [RFC3394], the AES Key Wrap Initialization Value **MUST** be set to 0xA6A6A6A6A6A6A6A6 (64 bits). The output of the AES Key Wrap operation is 192-bit long. The most significant 64-bit are copied in the One-Time Key Preamble field, while the 128 less significant bits are copied in the One-Time Key field of the LISP-SEC Authentication Data.

When decrypting an encrypted OTK the receiver **MUST** verify that the Initialization Value resulting from the AES Key Wrap decryption operation is equal to 0xA6A6A6A6A6A6A6A6. If this verification fails the receiver **MUST** discard the entire message.

When a 128-bit OTK is sent unencrypted the OTK Encryption ID is set to `NULL_KEY_WRAP_128`, and the OTK Preamble is set to `0x0000000000000000` (64 bits).

5.6. Map-Resolver Processing

Upon receiving an encapsulated Map-Request with the S-bit set, the Map-Resolver decapsulates the ECM message. The ITR-OTK, if encrypted, is decrypted as specified in Section 5.5.

Protecting the confidentiality of the ITR-OTK and, in general, the security of how the Map-Request is handed by the Map-Resolver to the Map-Server, is specific to the particular Mapping System used, and outside of the scope of this memo.

In Mapping Systems where the Map-Server is compliant with [RFC6833], the Map-Resolver originates a new ECM header with the S-bit set, that contains the unencrypted ITR-OTK, as specified in Section 5.5, and the other data derived from the ECM Authentication Data of the received encapsulated Map-Request.

The Map-Resolver then forwards to the Map-Server the received Map-Request, encapsulated in the new ECM header that includes the newly computed Authentication Data fields.

5.7. Map-Server Processing

Upon receiving an ECM encapsulated Map-Request with the S-bit set, the Map-Server process the Map-Request according to the value of the S-bit contained in the Map-Register sent by the ETR during registration.

If the S-bit contained in the Map-Register was clear the Map-Server decapsulates the ECM and generates a new ECM encapsulated Map-Request that does not contain an ECM Authentication Data, as specified in [RFC6830]. The Map-Server does not perform any further LISP-SEC processing, and the Map-Reply will not be protected.

If the S-bit contained in the Map-Register was set the Map-Server decapsulates the ECM and generates a new ECM Authentication Data. The Authentication Data includes the OTK-AD and the EID-AD, that contains EID-prefix authorization information, that are ultimately sent to the requesting ITR.

The Map-Server updates the OTK-AD by deriving a new OTK (MS-OTK) from the ITR-OTK received with the Map-Request. MS-OTK is derived applying the key derivation function specified in the KDF ID field. If the algorithm specified in the KDF ID field is not supported, the

Map-Server uses a different algorithm to derive the key and updates the KDF ID field accordingly.

The Map-Server and the ETR MUST be configured with a shared key for mapping registration according to [RFC6833]. If MS-OTK confidentiality is required, then the MS-OTK SHOULD be encrypted, by wrapping the MS-OTK with the algorithm specified by the OTK Encryption ID field as specified in Section 5.5.

The Map-Server includes in the EID-AD the longest match registered EID-prefix for the destination EID, and an HMAC of this EID-prefix. The HMAC is keyed with the ITR-OTK contained in the received ECM Authentication Data, and the HMAC algorithm is chosen according to the Requested HMAC ID field. If The Map-Server does not support this algorithm, the Map-Server uses a different algorithm and specifies it in the EID HMAC ID field. The scope of the HMAC operation covers the entire EID-AD, from the EID-AD Length field to the EID HMAC field, which must be set to 0 before the computation.

The Map-Server then forwards the updated ECM encapsulated Map-Request, that contains the OTK-AD, the EID-AD, and the received Map-Request to an authoritative ETR as specified in [RFC6830].

5.7.1. Map-Server Processing in Proxy mode

If the Map-Server is in proxy mode, it generates a Map-Reply, as specified in [RFC6830], with the S-bit set to 1. The Map-Reply includes the Authentication Data that contains the EID-AD, computed as specified in Section 5.7, as well as the PKT-AD computed as specified in Section 5.8.

5.8. ETR Processing

Upon receiving an ECM encapsulated Map-Request with the S-bit set, the ETR decapsulates the ECM message. The OTK field, if encrypted, is decrypted as specified in Section 5.5 to obtain the unencrypted MS-OTK.

The ETR then generates a Map-Reply as specified in [RFC6830] and includes the Authentication Data that contains the EID-AD, as received in the encapsulated Map-Request, as well as the PKT-AD.

The EID-AD is copied from the Authentication Data of the received encapsulated Map-Request.

The PKT-AD contains the HMAC of the whole Map-Reply packet, keyed with the MS-OTK and computed using the HMAC algorithm specified in the Requested HMAC ID field of the received encapsulated Map-Request.

If the ETR does not support the Requested HMAC ID, it uses a different algorithm and updates the PKT HMAC ID field accordingly. The scope of the HMAC operation covers the entire PKT-AD, from the Map-Reply Type field to the PKT HMAC field, which must be set to 0 before the computation.

Finally the ETR sends the Map-Reply to the requesting ITR as specified in [RFC6830].

6. Security Considerations

6.1. Mapping System Security

The LISP-SEC threat model described in Section 3, assumes that the LISP Mapping System is working properly and eventually delivers Map-Request messages to a Map-Server that is authoritative for the requested EID.

It is assumed that the Mapping System ensures the confidentiality of the OTK, and the integrity of the Map-Reply data. However, how the LISP Mapping System is secured is out of the scope of this document.

Similarly, Map-Register security, including the right for a LISP entity to register an EID-prefix or to claim presence at an RLOC, is out of the scope of LISP-SEC.

6.2. Random Number Generation

The ITR-OTK MUST be generated by a properly seeded pseudo-random (or strong random) source. See [RFC4086] for advice on generating security-sensitive random data

6.3. Map-Server and ETR Colocation

If the Map-Server and the ETR are colocated, LISP-SEC does not provide protection from overclaiming attacks mounted by the ETR. However, in this particular case, since the ETR is within the trust boundaries of the Map-Server, ETR's overclaiming attacks are not included in the threat model.

6.4. Deploying LISP-SEC

This memo is written according to [RFC2119]. Specifically, the use of the key word SHOULD "or the adjective 'RECOMMENDED', mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course".

Those deploying LISP-SEC according to this memo, should carefully weight how the LISP-SEC threat model applies to their particular use case or deployment. If they decide to ignore a particular recommendation, they should make sure the risk associated with the corresponding threats is well understood.

As an example, in certain closed and controlled deployments, it is possible that the threat associated with a MiTM between the xTR and the Mapping System is very low, and after carfeul consideration it may be decided to allow a NULL key wrapping algorithm while carrying the OTKs between the xTR and the Mapping System.

As an example at the other end of the spectrum, in certain other deployments, attackers may be very sophisticated, and force the deployers to enforce very strict policies in term of HMAC algorithms accepted by an ITR.

Similar considerations apply to the entire LISP-SEC threat model, and should guide the deployers and implementors whenever they encounter the key word SHOULD across this memo.

6.5. Shared Keys Provisioning

Provisioning of the keys shared between the ITR and the Map-Resolver as well as between the ETR and the Map-Server should be performed via an orchestration infrastructure and it is out of the scope of this draft. It is recommended that both shared keys are refreshed at periodical intervals to address key aging or attackers gaining unauthorized access to the shared keys. Shared keys should be unpredictable random values.

6.6. Replay Attacks

An attacker can capture a valid Map-Request and/or Map-Reply and replay it, however once the ITR receives the original Map-Reply the <nonce,ITR-OTK> pair stored at the ITR will be discarded. If a replayed Map-Reply arrives at the ITR, there is no <nonce,ITR-OTK> that matches the incoming Map-Reply and will be discarded.

In case of replayed Map-Request, the Map-Server, Map-Resolver and ETR will have to do a LISP-SEC computation. This is equivalent to a valid LISP-SEC computation and an attacker does not obtain any benefit.

6.7. Denial of Service and Distributed Denial of Service Attacks

LISP-SEC mitigates the risks of Denial of Service and Distributed Denial of Service attacks by protecting the integrity and authenticating the origin of the Map-Request/Map-Reply messages, and by preventing malicious ETRs from overclaiming EID prefixes that could re-direct traffic directed to a potentially large number of hosts.

7. IANA Considerations

7.1. ECM AD Type Registry

IANA is requested to create the "ECM Authentication Data Type" registry with values 0-255, for use in the ECM LISP-SEC Extensions Section 5.1. The registry MUST be initially populated with the following values:

Name	Value	Defined In
-----	-----	-----
Reserved	0	This memo
LISP-SEC-ECM-EXT	1	This memo

HMAC Functions

Values 2-255 are unassigned. They are to be assigned according to the "Specification Required" policy defined in [RFC5226].

7.2. Map-Reply AD Type Registry

IANA is requested to create the "Map-Reply Authentication Data Type" registry with values 0-255, for use in the Map-Reply LISP-SEC Extensions Section 5.2. The registry MUST be initially populated with the following values:

Name	Value	Defined In
-----	-----	-----
Reserved	0	This memo
LISP-SEC-MR-EXT	1	This memo

HMAC Functions

Values 2-255 are unassigned. They are to be assigned according to the "Specification Required" policy defined in [RFC5226].

7.3. HMAC Functions

IANA is requested to create the "LISP-SEC Authentication Data HMAC ID" registry with values 0-65535 for use as Requested HMAC ID, EID HMAC ID, and PKT HMAC ID in the LISP-SEC Authentication Data:

Name	Number	Defined In

NONE	0	This memo
AUTH-HMAC-SHA-1-96	1	[RFC2104]
AUTH-HMAC-SHA-256-128	2	[RFC6234]

HMAC Functions

Values 3-65535 are unassigned. They are to be assigned according to the "Specification Required" policy defined in [RFC5226].

AUTH-HMAC-SHA-1-96 MUST be supported, AUTH-HMAC-SHA-256-128 SHOULD be supported.

7.4. Key Wrap Functions

IANA is requested to create the "LISP-SEC Authentication Data Key Wrap ID" registry with values 0-65535 for use as OTK key wrap algorithms ID in the LISP-SEC Authentication Data:

Name	Number	Defined In

Reserved	0	This memo
NULL-KEY-WRAP-128	1	This memo
AES-KEY-WRAP-128	2	[RFC3394]

Key Wrap Functions

Values 3-65535 are unassigned. They are to be assigned according to the "Specification Required" policy defined in [RFC5226].

NULL-KEY-WRAP-128, and AES-KEY-WRAP-128 MUST be supported.

NULL-KEY-WRAP-128 is used to carry an unencrypted 128-bit OTK, with a 64-bit preamble set to 0x0000000000000000 (64 bits).

7.5. Key Derivation Functions

IANA is requested to create the "LISP-SEC Authentication Data Key Derivation Function ID" registry with values 0-65535 for use as KDF ID in the LISP-SEC Authentication Data:

Name	Number	Defined In

NONE	0	This memo
HKDF-SHA1-128	1	[RFC5869]

Key Derivation Functions

Values 2-65535 are unassigned. They are to be assigned according to the "Specification Required" policy defined in [RFC5226].

HKDF-SHA1-128 MUST be supported

8. Acknowledgements

The authors would like to acknowledge Pere Monclus, Dave Meyer, Dino Farinacci, Brian Weis, David McGrew, Darrel Lewis and Landon Curt Noll for their valuable suggestions provided during the preparation of this document.

9. Normative References

- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <<https://www.rfc-editor.org/info/rfc2104>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394, September 2002, <<https://www.rfc-editor.org/info/rfc3394>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.

- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.
- [RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <<https://www.rfc-editor.org/info/rfc5869>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, DOI 10.17487/RFC6833, January 2013, <<https://www.rfc-editor.org/info/rfc6833>>.
- [RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, DOI 10.17487/RFC6836, January 2013, <<https://www.rfc-editor.org/info/rfc6836>>.
- [RFC7835] Saucez, D., Iannone, L., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Threat Analysis", RFC 7835, DOI 10.17487/RFC7835, April 2016, <<https://www.rfc-editor.org/info/rfc7835>>.

Authors' Addresses

Fabio Maino
Cisco Systems
170 Tasman Drive
San Jose, California 95134
USA

Email: fmaino@cisco.com

Vina Ermagan
Cisco Systems
170 Tasman Drive
San Jose, California 95134
USA

Email: vermagan@cisco.com

Albert Cabellos
Universitat Politecnica de Catalunya
c/ Jordi Girona s/n
Barcelona 08034
Spain

Email: acabello@ac.upc.edu

Damien Saucez
INRIA
2004 route des Lucioles - BP 93
Sophia Antipolis
France

Email: damien.saucez@inria.fr

LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: January 4, 2018

V. Ermagan
A. Rodriguez-Natal
F. Coras
C. Moberg
R. Rahman
Cisco Systems
A. Cabellos-Aparicio
Technical University of Catalonia
F. Maino
Cisco Systems
July 3, 2017

LISP YANG Model
draft-ietf-lisp-yang-05

Abstract

This document describes a YANG data model to use with the Locator/ID Separation Protocol (LISP).

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, 2018.

Copyright Notice

Copyright (c) 2017 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
2. LISP Module	2
2.1. Module Structure	3
2.2. Module Definition	5
3. LISP-ITR Module	13
3.1. Module Structure	13
3.2. Module Definition	17
4. LISP-ETR Module	20
4.1. Module Structure	20
4.2. Module Definition	22
5. LISP-Map-Server Module	26
5.1. Module Structure	26
5.2. Module Definition	32
6. LISP-Map-Resolver Module	37
6.1. Module Structure	37
6.2. Module Definition	38
7. LISP-Address-Types Module	39
7.1. Module Definition	39
8. Acknowledgments	54
9. IANA Considerations	54
10. Security Considerations	54
11. Normative References	54
Authors' Addresses	55

1. Introduction

The Locator/ID Separation Protocol (LISP) defines several network elements subject to be configured. This document presents the YANG data models required for basic configuration of all major LISP [RFC6830] elements. The models also capture some essential operational data elements as well.

2. LISP Module

This module is the base LISP module that is augmented in multiple models to represent various LISP device roles.

2.1. Module Structure

```
module: ietf-lisp
```

```

  +--rw lisp
    +--rw locator-sets
      +--rw locator-set* [locator-set-name]
        +--rw locator-set-name      string
        +--rw (locator-type)?
          +--:(local-interface)
            +--rw interface* [interface-ref]
              +--rw interface-ref    if:interface-ref
              +--rw priority?        uint8
              +--rw weight?          uint8
              +--rw multicast-priority? uint8
              +--rw multicast-weight? uint8
          +--:(general-locator)
            +--rw locator* [id]
              +--rw id                string
              +--rw locator-address
                +--rw address-type    lisp-address-family-ref
                +--rw virtual-network-id? instance-id-type
                +--rw (address)?
                  +--:(no-address)
                    | +--rw no-address?      empty
                  +--:(ipv4)
                    | +--rw ipv4?            inet:ipv4-address
                  +--:(ipv4-prefix)
                    | +--rw ipv4-prefix?    inet:ipv4-prefix
                  +--:(ipv6)
                    | +--rw ipv6?            inet:ipv6-address
                  +--:(ipv6-prefix)
                    | +--rw ipv6-prefix?    inet:ipv6-prefix
                  +--:(mac)
                    | +--rw mac?            yang:mac-address
                  +--:(distinguished-name)
                    | +--rw distinguished-name? distinguished-name
                -type
                  +--:(as-number)
                    | +--rw as-number?      inet:as-number
                  +--:(null-address)
                    | +--rw null-address
                    |   +--rw address?      empty
                  +--:(afi-list)
                    | +--rw afi-list
                    |   +--rw address-list* simple-address
                  +--:(instance-id)
                    | +--rw instance-id
                    |   +--rw iid?          instance-id-type
                    |   +--rw mask-length?  uint8

```



```

|         +--rw address?          simple-address
+---: (as-number-lcaf)
|         +--rw as-number-lcaf
|         +--rw as?              inet:as-number
|         +--rw address?        simple-address
+---: (application-data)
|         +--rw application-data
|         +--rw address?          simple-address
|         +--rw protocol?         uint8
|         +--rw ip-tos?           int32
|         +--rw local-port-low?   inet:port-number
|         +--rw local-port-high?  inet:port-number
|         +--rw remote-port-low?  inet:port-number
|         +--rw remote-port-high? inet:port-number
+---: (geo-coordinates)
|         +--rw geo-coordinates
|         +--rw latitude?         bits
|         +--rw latitude-degrees? uint8
|         +--rw latitude-minutes? uint8
|         +--rw latitude-seconds? uint8
|         +--rw longitude?        bits
|         +--rw longitude-degrees? uint16
|         +--rw longitude-minutes? uint8
|         +--rw longitude-seconds? uint8
|         +--rw altitude?         int32
|         +--rw address?          simple-address
+---: (nat-traversal)
|         +--rw nat-traversal
|         +--rw ms-udp-port?      uint16
|         +--rw etr-udp-port?     uint16
|         +--rw global-etr-rloc?  simple-address
|         +--rw ms-rloc?          simple-address
|         +--rw private-etr-rloc? simple-address
|         +--rw rtr-rlocs*        simple-address
+---: (explicit-locator-path)
|         +--rw explicit-locator-path
|         +--rw hop* [hop-id]
|         +--rw hop-id           string
|         +--rw address?         simple-address
|         +--rw lrs-bits?        bits
+---: (source-dest-key)
|         +--rw source-dest-key
|         +--rw source?          simple-address
|         +--rw dest?            simple-address
+---: (key-value-address)
|         +--rw key-value-address
|         +--rw key?             simple-address
|         +--rw value?           simple-address

```

```

|                                     +--:(service-path)
|                                     +--rw service-path
|                                     +--rw service-path-id?  service-path-id-type
|                                     +--rw service-index?    uint8
|                                     +--rw priority?         uint8
|                                     +--rw weight?          uint8
|                                     +--rw multicast-priority? uint8
|                                     +--rw multicast-weight?  uint8
+--rw lisp-router-instances
  +--rw lisp-router-instance* [lisp-router-instance-id]
    +--rw lisp-router-instance-id  int32
    +--rw lisp-role* [lisp-role-type]
      | +--rw lisp-role-type    lisp-role-ref
    +--rw lisp-router-id
      +--rw site-id?  uint64
      +--rw xtr-id?   lisp:xtr-id-type

```

2.2. Module Definition

```

<CODE BEGINS> file "ietf-lisp@2017-07-01.yang"
module ietf-lisp {
  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp";
  prefix lisp;
  import ietf-interfaces {
    prefix if;
  }
  import ietf-lisp-address-types {
    prefix lacf;
  }
  import ietf-yang-types {
    prefix yang;
  }
  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "lisp@ietf.org";
  description
    "This YANG module defines the generic parameters for LISP.
    The module can be extended by vendors to define vendor-specific
    LISP parameters and policies."

```

Copyright (c) 2014 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions

Relating to IETF Documents
(<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC 6338; see
the RFC itself for full legal notices.

";

```
revision 2017-07-01 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6830";
}
identity lisp-role {
  description
    "LISP router role.";
}
identity itr {
  base lisp-role;
  description
    "LISP ITR.";
}
identity pitr {
  base lisp-role;
  description
    "LISP PITR.";
}
identity etr {
  base lisp-role;
  description
    "LISP ETR.";
}
identity petr {
  base lisp-role;
  description
    "LISP PETR.";
}
identity mapping-system {
  description
    "Mapping System interface";
}
identity single-node-mapping-system {
  base mapping-system;
  description
    "logically singular Map Server";
}
typedef mapping-system-ref {
  type identityref {
```

```
    base mapping-system;
  }
  description
    "Mapping System reference";
}

typedef lisp-role-ref {
  type identityref {
    base lisp-role;
  }
  description
    "LISP role reference";
}

typedef map-reply-action {
  type enumeration {
    enum no-action {
      value 0;
      description
        "Mapping is kept alive and no encapsulation occurs.";
    }
    enum natively-forward {
      value 1;
      description
        "Matching packets are not encapsulated or dropped but
        natively forwarded.";
    }
    enum send-map-request {
      value 2;
      description
        "Matching packets invoke Map-Requests.";
    }
    enum drop {
      value 3;
      description
        "Matching packets are dropped.";
    }
  }
  description
    "Defines the lisp map-cache ACT type";
  reference "https://tools.ietf.org/html/rfc6830#section-6.1.4";
}

typedef eid-id {
  type string;
  description
    "Type encoding of lisp-addresses to be generally used in EID
    keyed lists.";
}

typedef auth-key-type {
```

```
type enumeration {
  enum none {
    value 0;
    description
      "No authentication.";
  }
  enum hmac-sha-1-96 {
    value 1;
    description
      "HMAC-SHA-1-96 (RFC2404) authentication is used.";
  }
  enum hmac-sha-256-128 {
    value 2;
    description
      "HMAC-SHA-256-128 (RFC4868) authentication is used.";
  }
}
description
  "Enumeration of the authentication mechanisms supported by
  LISP.";
reference
  "https://tools.ietf.org/html/rfc6830#section-6.1.6";
}
typedef xtr-id-type {
  type binary {
    length "16";
  }
  description
    "128 bit xTR identifier.";
}

grouping locator-properties {
  description
    "Properties of a RLOC";
  leaf priority {
    type uint8;
    description
      "Locator priority.";
  }
  leaf weight {
    type uint8;
    description
      "Locator weight.";
  }
  leaf multicast-priority {
    type uint8;
    description
      "Locator's multicast priority";
  }
}
```

```
    }
    leaf multicast-weight {
      type uint8;
      description
        "Locator's multicast weight";
    }
  }

  grouping locators-grouping {
    description
      "Group that defines a list of LISP locators.";
    list locator {
      key "id";
      description
        "List of routing locators";
      leaf id {
        type string {
          length "1..64";
        }
        description
          "Locator id";
      }
      container locator-address {
        uses lcaf:lisp-address;
        description
          "The locator address provided in LISP canonincal
            address format.";
      }
      uses locator-properties;
    }
  }

  grouping local-locators-grouping {
    description
      "Group that defines a list of LISP locators.";
    list interface {
      key "interface-ref";
      description
        "The address type of the locator";
      leaf interface-ref {
        type if:interface-ref;
        description
          "The name of the interface supporting the locator.";
      }
      uses locator-properties;
    }
  }
}
```

```
grouping mapping {
  description
    "Group that defines a LISP mapping.";
  container eid {
    uses lcaf:lisp-address;
    description
      "End-host Identifier (EID) to be mapped to a list of
        locators";
  }
  leaf time-to-live {
    type uint32;
    units minutes;
    description
      "Mapping validity period in minutes.";
  }
  leaf creation-time {
    type yang:date-and-time;
    description
      "Time when the mapping was created.";
  }
  leaf authoritative {
    type bits {
      bit A {
        description
          "Authoritative bit.";
      }
    }
    description
      "Bit that indicates if mapping comes from an
        authoritative source.";
  }
  leaf static {
    type boolean;
    default "false";
    description
      "This leaf should be true if the mapping is static.";
  }
  choice locator-list {
    description
      "list of locartors are either negative, or positive.";
    case negative-mapping {
      leaf map-reply-action {
        type map-reply-action;
        description
          "Forwarding action for a negative mapping.";
      }
    }
    case positive-mapping {
```

```
        container rlocs {
            uses locators-grouping;
            description
                "List of locators for a positive mapping.";
        }
    }
}

grouping mappings {
    description
        "Group that defines a list of LISP mappings.";
    list virtual-network {
        key "vni";
        description
            "Virtual network to which the mappings belong.";
        leaf vni {
            type lcaf:instance-id-type;
            description
                "Virtual network identifier.";
        }
        container mappings {
            description
                "Mappings within the virtual network.";
            list mapping {
                key "id";
                description
                    "List of EID to RLOCs mappings.";
                leaf id {
                    type eid-id;
                    description
                        "Id that uniquely identifies a mapping.";
                }
                uses mapping;
            }
        }
    }
}

container lisp {
    description
        "Parameters for the LISP subsystem.";

    container locator-sets {
        description
            "Container that defines a named locator set which can be
            referenced elsewhere.";
        list locator-set {
```



```
    key "locator-set-name";
    description
        "Multiple locator sets can be defined.";
    leaf locator-set-name {
        type string {
            length "1..64";
        }
        description
            "Locator set name";
    }
    choice locator-type {
        description
            "Locator sets can be based on local interfaces, or
            general locators.";
        case local-interface {
            uses local-locators-grouping;
            description
                "List of locators in this set based on local
                interfaces.";
        }
        case general-locator {
            uses locators-grouping;
            description
                "List of locators in this set based on lisp-address.";
        }
    }
}

container lisp-router-instances {
    description
        "Different LISP routers instantiated in the device";
    list lisp-router-instance {
        key "lisp-router-instance-id";
        description
            "Each entry contains parameters for a LISP router.";
        leaf lisp-router-instance-id {
            type int32;
            description
                "Arbitrary lisp-router id.";
        }
        list lisp-role {
            key lisp-role-type;
            description
                "List of lisp device roles such as MS, MR, ITR,
                Pitr, ETR or PETR.";
            leaf lisp-role-type {
                type lisp-role-ref;
            }
        }
    }
}
```

```

        description
            "The type of LISP device - identity derived from the
             'lisp-device' base identity.";
    }
}
container lisp-router-id {
    when "../lisp-role/lisp-role-type = 'itr' or
         ../lisp-role/lisp-role-type = 'pitrr' or
         ../lisp-role/lisp-role-type = 'etr' or
         ../lisp-role/lisp-role-type = 'petrr'" {
        description "Only when ITR, PITR, ETR or PETR.";
    }
    description
        "Site-ID and xTR-ID of the device.";
    leaf site-id {
        type uint64;
        description "Site ID";
    }
    leaf xtr-id {
        type lisp:xtr-id-type;
        description "xTR ID";
    }
}
}
}
}
<CODE ENDS>
```

3. LISP-ITR Module

This module captures the configuration data model of a LISP ITR. The model also captures some operational data elements.

3.1. Module Structure

```

module: ietf-lisp-itr
augment /lisp:lisp/lisp:lisp-router-instances/lisp:lisp-router-instance:
  +--rw itr!
    +--rw rloc-probing!
      |   +--rw interval?          uint16
      |   +--rw retries?           uint8
      |   +--rw retries-interval?  uint16
    +--rw itr-rlocs?      -> /lisp:lisp/locator-sets/locator-set/locator-set
-name
  +--rw map-resolvers
    |   +--rw map-resolver*    inet:ip-address
  +--rw proxy-etrs
    |   +--rw proxy-etr-address*  inet:ip-address

```

```

+--rw map-cache
  +--rw virtual-network* [vni]
    +--rw vni          lcaf:instance-id-type
    +--rw mappings
      +--rw mapping* [id]
        +--rw id          eid-id
        +--rw eid
          +--rw address-type          lisp-address-family-ref
          +--rw virtual-network-id?   instance-id-type
          +--rw (address)?
            +--:(no-address)
              | +--rw no-address?          empty
            +--:(ipv4)
              | +--rw ipv4?                inet:ipv4-address
            +--:(ipv4-prefix)
              | +--rw ipv4-prefix?         inet:ipv4-prefix
            +--:(ipv6)
              | +--rw ipv6?                inet:ipv6-address
            +--:(ipv6-prefix)
              | +--rw ipv6-prefix?         inet:ipv6-prefix
            +--:(mac)
              | +--rw mac?                 yang:mac-address
            +--:(distinguished-name)
              | +--rw distinguished-name?   distinguished-name-ty
          +--:(as-number)
            | +--rw as-number?              inet:as-number
          +--:(null-address)
            | +--rw null-address
            |   +--rw address?          empty
          +--:(afi-list)
            | +--rw afi-list
            |   +--rw address-list*     simple-address
          +--:(instance-id)
            | +--rw instance-id
            |   +--rw iid?              instance-id-type
            |   +--rw mask-length?     uint8
            |   +--rw address?         simple-address
          +--:(as-number-lcaf)
            | +--rw as-number-lcaf
            |   +--rw as?              inet:as-number
            |   +--rw address?         simple-address
          +--:(application-data)
            | +--rw application-data
            |   +--rw address?          simple-address
            |   +--rw protocol?        uint8
            |   +--rw ip-tos?          int32
            |   +--rw local-port-low?   inet:port-number
            |   +--rw local-port-high?  inet:port-number

```

pe

```

|         +--rw remote-port-low?      inet:port-number
|         +--rw remote-port-high?     inet:port-number
+---:(geo-coordinates)
|   +--rw geo-coordinates
|   |   +--rw latitude?                bits
|   |   +--rw latitude-degrees?       uint8
|   |   +--rw latitude-minutes?       uint8
|   |   +--rw latitude-seconds?       uint8
|   |   +--rw longitude?              bits
|   |   +--rw longitude-degrees?      uint16
|   |   +--rw longitude-minutes?      uint8
|   |   +--rw longitude-seconds?      uint8
|   |   +--rw altitude?               int32
|   |   +--rw address?                simple-address
+---:(nat-traversal)
|   +--rw nat-traversal
|   |   +--rw ms-udp-port?             uint16
|   |   +--rw etr-udp-port?           uint16
|   |   +--rw global-etr-rloc?        simple-address
|   |   +--rw ms-rloc?                simple-address
|   |   +--rw private-etr-rloc?       simple-address
|   |   +--rw rtr-rlocs*              simple-address
+---:(explicit-locator-path)
|   +--rw explicit-locator-path
|   |   +--rw hop* [hop-id]
|   |   |   +--rw hop-id              string
|   |   |   +--rw address?            simple-address
|   |   |   +--rw lrs-bits?           bits
+---:(source-dest-key)
|   +--rw source-dest-key
|   |   +--rw source?                  simple-address
|   |   +--rw dest?                   simple-address
+---:(key-value-address)
|   +--rw key-value-address
|   |   +--rw key?                     simple-address
|   |   +--rw value?                   simple-address
+---:(service-path)
|   +--rw service-path
|   |   +--rw service-path-id?         service-path-id-type
|   |   +--rw service-index?          uint8
+--rw time-to-live?                    uint32
+--rw creation-time?                   yang:date-and-time
+--rw authoritative?                   bits
+--rw static?                          boolean
+--rw (locator-list)?
|   +---:(negative-mapping)
|   |   +--rw map-reply-action?       map-reply-action
|   +---:(positive-mapping)

```

```

+--rw rlocs
  +--rw locator* [id]
    +--rw id string
    +--rw locator-address
      | +--rw address-type lisp-address-fa
mily-ref
      | +--rw virtual-network-id? instance-id-ty
e
      | +--rw (address)?
      |   +--:(no-address)
      |     | +--rw no-address? empty
      |   +--:(ipv4)
      |     | +--rw ipv4? inet:ipv4
-address
      |   +--:(ipv4-prefix)
      |     | +--rw ipv4-prefix? inet:ipv4
-prefix
      |   +--:(ipv6)
      |     | +--rw ipv6? inet:ipv6
-address
      |   +--:(ipv6-prefix)
      |     | +--rw ipv6-prefix? inet:ipv6
-prefix
      |   +--:(mac)
      |     | +--rw mac? yang:mac-
address
      |   +--:(distinguished-name)
      |     | +--rw distinguished-name? distingui
shed-name-type
      |   +--:(as-number)
      |     | +--rw as-number? inet:as-n
umber
      |   +--:(null-address)
      |     | +--rw null-address
      |     | +--rw address? empty
      |   +--:(afi-list)
      |     | +--rw afi-list
      |     | +--rw address-list* simple-address
      |   +--:(instance-id)
      |     | +--rw instance-id
      |     |   +--rw iid? instance-id-type
      |     |   +--rw mask-length? uint8
      |     |   +--rw address? simple-address
      |   +--:(as-number-lcaf)
      |     | +--rw as-number-lcaf
      |     |   +--rw as? inet:as-number
      |     |   +--rw address? simple-address
      |   +--:(application-data)
      |     | +--rw application-data
      |     |   +--rw address? simple-addr
ess
      |   +--rw protocol? uint8
      |   +--rw ip-tos? int32
      |   +--rw local-port-low? inet:port-n
umber
      |   +--rw local-port-high? inet:port-n
umber
      |   +--rw remote-port-low? inet:port-n
umber
      |   +--rw remote-port-high? inet:port-n
umber
      |   +--:(geo-coordinates)

```


			+--rw geo-coordinates	
			+--rw latitude?	bits
			+--rw latitude-degrees?	uint8
			+--rw latitude-minutes?	uint8
			+--rw latitude-seconds?	uint8
			+--rw longitude?	bits
			+--rw longitude-degrees?	uint16
			+--rw longitude-minutes?	uint8
			+--rw longitude-seconds?	uint8
			+--rw altitude?	int32
			+--rw address?	simple-add
ress			+---:(nat-traversal)	
			+--rw nat-traversal	
			+--rw ms-udp-port?	uint16
			+--rw etr-udp-port?	uint16
			+--rw global-etr-rloc?	simple-addr
ess			+--rw ms-rloc?	simple-addr
ess			+--rw private-etr-rloc?	simple-addr
ess			+--rw rtr-rlocs*	simple-addr
ess			+---:(explicit-locator-path)	
			+--rw explicit-locator-path	
			+--rw hop* [hop-id]	
			+--rw hop-id	string
			+--rw address?	simple-address
			+--rw lrs-bits?	bits
			+---:(source-dest-key)	
			+--rw source-dest-key	
			+--rw source?	simple-address
			+--rw dest?	simple-address
			+---:(key-value-address)	
			+--rw key-value-address	
			+--rw key?	simple-address
			+--rw value?	simple-address
			+---:(service-path)	
			+--rw service-path	
			+--rw service-path-id?	service-path
-id-type			+--rw service-index?	uint8
			+--rw priority?	uint8
			+--rw weight?	uint8
			+--rw multicast-priority?	uint8
			+--rw multicast-weight?	uint8

3.2. Module Definition

```

<CODE BEGINS> file "ietf-lisp-itr@2017-07-01.yang"
module ietf-lisp-itr {
  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-itr";
  prefix lisp-itr;

```

```
import ietf-lisp {
  prefix lisp;
}
import ietf-inet-types {
  prefix inet;
}
organization
  "IETF LISP (Locator/ID Separation Protocol) Working Group";
contact
  "lisp@ietf.org";
description
  "This YANG module defines the generic parameters for a LISP
  ITR. The module can be extended by vendors to define
  vendor-specific parameters and policies."

  Copyright (c) 2015 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC 6338; see
  the RFC itself for full legal notices.
";
revision 2017-07-01 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6830";
}
augment "/lisp:lisp/lisp:lisp-router-instances/lisp:lisp-router-instance" {
  when "lisp:lisp-role/lisp:lisp-role-type = 'lisp:itr' or
    lisp:lisp-role/lisp:lisp-role-type = 'lisp:pitr'" {
    description
      "Augment is valid when LISP role type is ITR or PITR.";
  }
  description
    "This augments LISP devices list with (P)ITR specific
    parameters.";
  container itr {
    presence "LISP (P)ITR operation enabled";
    description
      "ITR parameters";
    container rloc-probing {
```



```
presence "RLOC probing active";
description
  "RLOC-probing parameters";
leaf interval {
  type uint16;
  units "seconds";
  description
    "Interval in seconds for resending the probes";
}
leaf retries {
  type uint8;
  description
    "Number of retries for sending the probes";
}
leaf retries-interval {
  type uint16;
  units "seconds";
  description
    "Interval in seconds between retries when sending probes.
    The action taken if all retries fail to receive is
    implementation specific.";
}
}
leaf itr-rlocs {
  type leafref {
    path "/lisp:lisp/lisp:locator-sets/lisp:locator-set/"
      + "lisp:locator-set-name";
  }
  description
    "Reference to a locator set that the (P)ITR includes in
    Map-Requests";
}
container map-resolvers {
  description
    "Map-Resolvers that the (P)ITR uses.";
  leaf-list map-resolver {
    type inet:ip-address;
    min-elements 1;
    description
      "Each Map-Resolver within the list of Map-Resolvers.";
  }
}
container proxy-etrs {
  when "../lisp:lisp-role/lisp:lisp-role-type = 'lisp:itr'" {
    description
      "Container exists only when LISP role type is ITR";
  }
  description

```

```

        "Proxy ETRs that the ITR uses.";
    leaf-list proxy-etr-address{
        type inet:ip-address;
        description
            "Proxy ETR RLOC address.";
    }
}
container map-cache{
    uses lisp:mappings;
    description
        "EID to RLOCs mappings cache.";
}
}
}
<CODE ENDS>

```

4. LISP-ETR Module

This module captures the configuration data model of a LISP ETR. The model also captures some operational data elements.

4.1. Module Structure

```

module: ietf-lisp-etr
augment /lisp:lisp/lisp:lisp-router-instances/lisp:lisp-router-instance:
  +--rw etr!
    +--rw map-servers
      |   +--rw map-server* [ms-address]
      |   |   +--rw ms-address          inet:ip-address
      |   |   +--rw auth-key?           string
      |   |   +--rw auth-key-type?      lisp:auth-key-type
    +--rw local-eids
      |   +--rw virtual-network* [vni]
      |   |   +--rw vni                 lcaf:instance-id-type
      |   |   +--rw eids
      |   |   |   +--rw local-eid* [id]
      |   |   |   |   +--rw id                                lisp:eid-id
      |   |   |   |   +--rw eid-address
      |   |   |   |   |   +--rw address-type                lisp-address-family-ref
      |   |   |   |   |   +--rw virtual-network-id?         instance-id-type
      |   |   |   |   |   +--rw (address)?
      |   |   |   |   |   |   +--:(no-address)
      |   |   |   |   |   |   |   +--rw no-address?         empty
      |   |   |   |   |   |   |   +--:(ipv4)
      |   |   |   |   |   |   |   |   +--rw ipv4?           inet:ipv4-address
      |   |   |   |   |   |   |   |   +--:(ipv4-prefix)
      |   |   |   |   |   |   |   |   |   +--rw ipv4-prefix? inet:ipv4-prefix

```

pe

```

+---:(ipv6)
|   +---rw ipv6?                               inet:ipv6-address
+---:(ipv6-prefix)
|   +---rw ipv6-prefix?                         inet:ipv6-prefix
+---:(mac)
|   +---rw mac?                                yang:mac-address
+---:(distinguished-name)
|   +---rw distinguished-name?                  distinguished-name-ty

+---:(as-number)
|   +---rw as-number?                           inet:as-number
+---:(null-address)
|   +---rw null-address
|       +---rw address?    empty
+---:(afi-list)
|   +---rw afi-list
|       +---rw address-list*    simple-address
+---:(instance-id)
|   +---rw instance-id
|       +---rw iid?              instance-id-type
|       +---rw mask-length?      uint8
|       +---rw address?          simple-address
+---:(as-number-lcaf)
|   +---rw as-number-lcaf
|       +---rw as?                inet:as-number
|       +---rw address?          simple-address
+---:(application-data)
|   +---rw application-data
|       +---rw address?            simple-address
|       +---rw protocol?           uint8
|       +---rw ip-tos?             int32
|       +---rw local-port-low?     inet:port-number
|       +---rw local-port-high?    inet:port-number
|       +---rw remote-port-low?    inet:port-number
|       +---rw remote-port-high?   inet:port-number
+---:(geo-coordinates)
|   +---rw geo-coordinates
|       +---rw latitude?           bits
|       +---rw latitude-degrees?   uint8
|       +---rw latitude-minutes?   uint8
|       +---rw latitude-seconds?   uint8
|       +---rw longitude?          bits
|       +---rw longitude-degrees?   uint16
|       +---rw longitude-minutes?   uint8
|       +---rw longitude-seconds?   uint8
|       +---rw altitude?           int32
|       +---rw address?            simple-address
+---:(nat-traversal)
|   +---rw nat-traversal

```

```

    +--rw ms-udp-port?          uint16
    +--rw etr-udp-port?         uint16
    +--rw global-etr-rloc?      simple-address
    +--rw ms-rloc?              simple-address
    +--rw private-etr-rloc?     simple-address
    +--rw rtr-rlocs*            simple-address
  +---:(explicit-locator-path)
    +--rw explicit-locator-path
    +--rw hop* [hop-id]
      +--rw hop-id              string
      +--rw address?            simple-address
      +--rw lrs-bits?           bits
  +---:(source-dest-key)
    +--rw source-dest-key
      +--rw source?             simple-address
      +--rw dest?               simple-address
  +---:(key-value-address)
    +--rw key-value-address
      +--rw key?                simple-address
      +--rw value?              simple-address
  +---:(service-path)
    +--rw service-path
      +--rw service-path-id?    service-path-id-type
      +--rw service-index?      uint8
  +--rw rlocs?                  -> /lisp:lisp/locator-sets/loc
ator-set/locator-set-name
  +--rw record-ttl?              uint32
  +--rw want-map-notify?         boolean
  +--rw proxy-reply?             boolean
  +--rw registration-interval?   uint16

```

4.2. Module Definition

```

<CODE BEGINS> file "ietf-lisp-etr@2017-07-01.yang"
module ietf-lisp-etr {
  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-etr";
  prefix lisp-etr;
  import ietf-lisp {
    prefix lisp;
  }
  import ietf-lisp-address-types {
    prefix lacf;
  }
  import ietf-inet-types {
    prefix inet;
  }
  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact

```

```
"lisp@ietf.org";
description
  "This YANG module defines the generic parameters for a LISP
  ETR. The module can be extended by vendors to define
  vendor-specific parameters and policies.

  Copyright (c) 2015 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC 6338; see
  the RFC itself for full legal notices.
";
revision 2017-07-01 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6830";
}
augment "/lisp:lisp/lisp:lisp-router-instances/lisp:lisp-router-instance" {
  when "lisp:lisp-role/lisp:lisp-role-type = 'lisp:etr' or
        lisp:lisp-role/lisp:lisp-role-type = 'lisp:petr'" {
    description
      "Augment is valid when LISP device type is (P)ETR.";
  }
  description
    "This augments LISP devices list with (P)ETR specific
    parameters.";
  container etr {
    presence "LISP (P)ETR operation enabled";
    description
      "(P)ETR parameters.";

    container map-servers {
      when "../lisp:lisp-role/lisp:lisp-role-type = 'lisp:etr'" {
        description
          "Container exists only when LISP device type is ETR.";
      }
      description
        "Map-Servers that the ETR uses.";
      list map-server {
        key "ms-address";
```

```
    description
      "Each Map-Server within the list of Map-Servers.";
    leaf ms-address {
      type inet:ip-address;
      description
        "Map-Server address.";
    }
    leaf auth-key {
      type string;
      description
        "Map-Server authentication key.";
    }
    leaf auth-key-type {
      type lisp:auth-key-type;
      description
        "Map-Server authentication type.";
    }
  }
}

container local-eids {
  when "../lisp:lisp-role/lisp:lisp-role-type = 'lisp:etr'" {
    description
      "Container exists only when LISP device type is ETR.";
  }
  description
    "Virtual networks served by the ETR.";
  list virtual-network {
    key "vni";
    description
      "Virtual network for local-EIDs.";
    leaf vni {
      type lisp:instance-id-type;
      description
        "Virtual network identifier.";
    }
  }
  container eids {
    description
      "EIDs served by the ETR.";
    list local-eid {
      key "id";
      min-elements 1;
      description
        "List of local EIDs.";
      leaf id {
        type lisp:eid-id;
        description
          "Unique id of local EID.";
      }
    }
  }
}
```

```
    }
    container eid-address {
      uses lcaf:lisp-address;
      description
        "EID address in generic LISP address format.";
    }
    leaf rlocs {
      type leafref {
        path "/lisp:lisp/lisp:locator-sets/lisp:locator-set/"
          + "lisp:locator-set-name";
      }
      description
        "Locator set mapped to this local EID.";
    }
    leaf record-ttl {
      type uint32;
      units minutes;
      description
        "Validity period of the EID to RLOCs mapping provided
        in Map-Replies.";
    }
    leaf want-map-notify {
      type boolean;
      default "true";
      description
        "Flag which if set in a Map-Register requests that a
        Map-Notify be sent in response.";
    }
    leaf proxy-reply {
      type boolean;
      default "false";
      description
        "Flag which if set in a Map-Register requests that the
        Map-Server proxy Map-Replies for the ETR.";
    }
    leaf registration-interval {
      type uint16;
      units "seconds";
      default "60";
      description
        "Interval between consecutive Map-Register messages.";
    }
  }
}
}
}
}
```

```

}
<CODE ENDS>

```

5. LISP-Map-Server Module

This module captures the configuration data model of a LISP Map Server [RFC6833]. The model also captures some operational data elements.

5.1. Module Structure

```

module: ietf-lisp-mapserver
augment /lisp:lisp/lisp-router-instances/lisp:lisp-router-instance:
  +--rw map-server!
    +--rw sites
      +--rw site* [site-id]
        +--rw site-id      uint64
        +--rw auth-key
          +--rw auth-key-value?  string
          +--rw auth-key-type*   lisp:auth-key-type
    +--rw virtual-network-ids
      +--rw virtual-network-identifier* [vni]
        +--rw vni              lcaf:instance-id-type
        +--rw mappings
          +--rw mapping* [eid-id]
            +--rw eid-id              lisp:eid-id
            +--rw eid-address
              +--rw address-type      lisp-address-family-ref
              +--rw virtual-network-id? instance-id-type
              +--rw (address)?
                +--:(no-address)
                | +--rw no-address?      empty
                +--:(ipv4)
                | +--rw ipv4?            inet:ipv4-address
                +--:(ipv4-prefix)
                | +--rw ipv4-prefix?    inet:ipv4-prefix
                +--:(ipv6)
                | +--rw ipv6?            inet:ipv6-address
                +--:(ipv6-prefix)
                | +--rw ipv6-prefix?    inet:ipv6-prefix
                +--:(mac)
                | +--rw mac?            yang:mac-address
                +--:(distinguished-name)
                | +--rw distinguished-name? distinguished-name-ty
                +--:(as-number)
                | +--rw as-number?      inet:as-number
                +--:(null-address)
                | +--rw null-address

```



```

|         +--rw address?    empty
+---:(afi-list)
|         +--rw afi-list
|         +--rw address-list*  simple-address
+---:(instance-id)
|         +--rw instance-id
|         +--rw iid?          instance-id-type
|         +--rw mask-length?  uint8
|         +--rw address?      simple-address
+---:(as-number-lcaf)
|         +--rw as-number-lcaf
|         +--rw as?          inet:as-number
|         +--rw address?     simple-address
+---:(application-data)
|         +--rw application-data
|         +--rw address?      simple-address
|         +--rw protocol?     uint8
|         +--rw ip-tos?       int32
|         +--rw local-port-low?  inet:port-number
|         +--rw local-port-high? inet:port-number
|         +--rw remote-port-low? inet:port-number
|         +--rw remote-port-high? inet:port-number
+---:(geo-coordinates)
|         +--rw geo-coordinates
|         +--rw latitude?      bits
|         +--rw latitude-degrees? uint8
|         +--rw latitude-minutes? uint8
|         +--rw latitude-seconds? uint8
|         +--rw longitude?     bits
|         +--rw longitude-degrees? uint16
|         +--rw longitude-minutes? uint8
|         +--rw longitude-seconds? uint8
|         +--rw altitude?      int32
|         +--rw address?      simple-address
+---:(nat-traversal)
|         +--rw nat-traversal
|         +--rw ms-udp-port?    uint16
|         +--rw etr-udp-port?   uint16
|         +--rw global-etr-rloc? simple-address
|         +--rw ms-rloc?        simple-address
|         +--rw private-etr-rloc? simple-address
|         +--rw rtr-rlocs*      simple-address
+---:(explicit-locator-path)
|         +--rw explicit-locator-path
|         +--rw hop* [hop-id]
|         +--rw hop-id         string
|         +--rw address?       simple-address
|         +--rw lrs-bits?      bits

```

			<pre> +---:(source-dest-key) +---rw source-dest-key +---rw source? simple-address +---rw dest? simple-address +---:(key-value-address) +---rw key-value-address +---rw key? simple-address +---rw value? simple-address +---:(service-path) +---rw service-path +---rw service-path-id? service-path-id-type +---rw service-index? uint8 +---rw site-id* uint64 +---rw more-specifics-accepted? boolean +---rw mapping-expiration-timeout? int16 +---rw mapping-records +---rw mapping-record* [xtr-id] +---rw xtr-id lisp:xtr-id-type +---rw site-id? uint64 +---rw eid +---rw address-type lisp-address-family-r </pre>
ef			<pre> +---rw virtual-network-id? instance-id-type +---rw (address)? +---:(no-address) +---rw no-address? empty +---:(ipv4) +---rw ipv4? inet:ipv4-addre </pre>
ss			
x			<pre> +---:(ipv4-prefix) +---rw ipv4-prefix? inet:ipv4-prefi </pre>
ss			
x			<pre> +---:(ipv6) +---rw ipv6? inet:ipv6-addre </pre>
ss			
x			<pre> +---:(ipv6-prefix) +---rw ipv6-prefix? inet:ipv6-prefi </pre>
s			<pre> +---:(mac) +---rw mac? yang:mac-addres </pre>
ame-type			<pre> +---:(distinguished-name) +---rw distinguished-name? distinguished-n </pre>
			<pre> +---:(as-number) +---rw as-number? inet:as-number +---:(null-address) +---rw null-address +---rw address? empty +---:(afi-list) +---rw afi-list +---rw address-list* simple-address +---:(instance-id) +---rw instance-id +---rw iid? instance-id-type </pre>

```

    +--rw mask-length?    uint8
    +--rw address?        simple-address
+--:(as-number-lcaf)
  +--rw as-number-lcaf
    +--rw as?            inet:as-number
    +--rw address?      simple-address
+--:(application-data)
  +--rw application-data
    +--rw address?      simple-address
    +--rw protocol?    uint8
    +--rw ip-tos?      int32
    +--rw local-port-low?  inet:port-number
    +--rw local-port-high? inet:port-number
    +--rw remote-port-low? inet:port-number
    +--rw remote-port-high? inet:port-number
+--:(geo-coordinates)
  +--rw geo-coordinates
    +--rw latitude?      bits
    +--rw latitude-degrees? uint8
    +--rw latitude-minutes? uint8
    +--rw latitude-seconds? uint8
    +--rw longitude?     bits
    +--rw longitude-degrees? uint16
    +--rw longitude-minutes? uint8
    +--rw longitude-seconds? uint8
    +--rw altitude?     int32
    +--rw address?      simple-address
+--:(nat-traversal)
  +--rw nat-traversal
    +--rw ms-udp-port?    uint16
    +--rw etr-udp-port?   uint16
    +--rw global-etr-rloc? simple-address
    +--rw ms-rloc?        simple-address
    +--rw private-etr-rloc? simple-address
    +--rw rtr-rlocs*      simple-address
+--:(explicit-locator-path)
  +--rw explicit-locator-path
    +--rw hop* [hop-id]
      +--rw hop-id    string
      +--rw address?  simple-address
      +--rw lrs-bits? bits
+--:(source-dest-key)
  +--rw source-dest-key
    +--rw source?    simple-address
    +--rw dest?      simple-address
+--:(key-value-address)
  +--rw key-value-address
    +--rw key?       simple-address

```

				+--rw value?	simple-address
				+++:(service-path)	
				+--rw service-path	
				+--rw service-path-id?	service-path-id-ty
pe				+--rw service-index?	uint8
				+--rw time-to-live?	uint32
				+--rw creation-time?	yang:date-and-time
				+--rw authoritative?	bits
				+--rw static?	boolean
				+--rw (locator-list)?	
				+++:(negative-mapping)	
				+--rw map-reply-action?	map-reply-action
				+++:(positive-mapping)	
				+--rw rlocs	
				+--rw locator* [id]	
				+--rw id	string
				+--rw locator-address	
ess-family-ref				+--rw address-type	lisp-addr
id-type				+--rw virtual-network-id?	instance-
				+--rw (address)?	
				+++:(no-address)	
ty				+--rw no-address?	emp
				+++:(ipv4)	
t:ipv4-address				+--rw ipv4?	ine
				+++:(ipv4-prefix)	
t:ipv4-prefix				+--rw ipv4-prefix?	ine
				+++:(ipv6)	
t:ipv6-address				+--rw ipv6?	ine
				+++:(ipv6-prefix)	
t:ipv6-prefix				+--rw ipv6-prefix?	ine
				+++:(mac)	
g:mac-address				+--rw mac?	yan
				+++:(distinguished-name)	
tinguished-name-type				+--rw distinguished-name?	dis
				+++:(as-number)	
t:as-number				+--rw as-number?	ine
				+++:(null-address)	
				+--rw null-address	
				+--rw address?	empty
				+++:(afi-list)	
				+--rw afi-list	
dress				+--rw address-list*	simple-ad
				+++:(instance-id)	
				+--rw instance-id	
d-type				+--rw iid?	instance-i
				+--rw mask-length?	uint8
ress				+--rw address?	simple-add
				+++:(as-number-lcaf)	

			+--rw as-number-lcaf	
			+--rw as? inet:as-number	
			+--rw address? simple-address	
			+++:(application-data)	
			+--rw application-data	
e-address			+--rw address?	simpl
			+--rw protocol?	uint8
port-number			+--rw ip-tos?	int32
port-number			+--rw local-port-low?	inet:
port-number			+--rw local-port-high?	inet:
port-number			+--rw remote-port-low?	inet:
port-number			+--rw remote-port-high?	inet:
			+++:(geo-coordinates)	
			+--rw geo-coordinates	
8			+--rw latitude?	bits
			+--rw latitude-degrees?	uint
8			+--rw latitude-minutes?	uint
			+--rw latitude-seconds?	uint
8			+--rw longitude?	bits
			+--rw longitude-degrees?	uint
16			+--rw longitude-minutes?	uint
8			+--rw longitude-seconds?	uint
8			+--rw altitude?	int3
2			+--rw address?	simp
le-address			+++:(nat-traversal)	
			+--rw nat-traversal	
6			+--rw ms-udp-port?	uint1
			+--rw etr-udp-port?	uint1
6			+--rw global-etr-rloc?	simpl
e-address			+--rw ms-rloc?	simpl
e-address			+--rw private-etr-rloc?	simpl
e-address			+--rw rtr-rlocs*	simpl
e-address			+++:(explicit locator-path)	
			+--rw explicit-locator-path	
			+--rw hop* [hop-id]	
			+--rw hop-id	string
ress			+--rw address?	simple-add
			+--rw lrs-bits?	bits
			+++:(source-dest-key)	
			+--rw source-dest-key	
			+--rw source?	simple-address
			+--rw dest?	simple-address
			+++:(key-value-address)	

						<pre> +--rw key-value-address +--rw key? simple-address +--rw value? simple-address +--:(service-path) +--rw service-path </pre>
--	--	--	--	--	--	--

```
e-path-id-type |
|
|
|
|
|
|--ro counters
|   |--ro map-registers-in?          yang:counter32
|   |--ro map-registers-in-auth-failed? yang:counter32
|   |--ro map-notify-records-out?     yang:counter32
|   |--ro proxy-reply-records-out?    yang:counter32
|   |--ro map-requests-forwarded-out? yang:counter32
|--rw mapping-system-type? lisp:mapping-system-ref
--ro summary
|   |--ro number-configured-sites?    uint32
|   |--ro number-registered-sites?    uint32
|   --ro af-datum
|       |--ro af-data* [address-type]
|           |--ro address-type        lcac:lisp-address-family-ref
|           |--ro number-configured-eids? uint32
|           |--ro number-registered-eids? uint32
--ro counters
|   |--ro map-registers-in?          yang:counter32
|   |--ro map-registers-in-auth-failed? yang:counter32
|   |--ro map-notify-records-out?     yang:counter32
|   |--ro proxy-reply-records-out?    yang:counter32
|   |--ro map-requests-forwarded-out? yang:counter32
```

5.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp-mapserver@2017-07-01.yang"
module ietf-lisp-mapserver {
  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver";
  prefix lisp-ms;
  import ietf-lisp {
    prefix lisp;
  }
  import ietf-lisp-address-types {
    prefix lacf;
  }
  import ietf-yang-types {
    prefix yang;
    revision-date 2013-07-15;
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
```



```
"lisp@ietf.org";
description
  "This YANG module defines the generic parameters for a LISP
  Map-Server. The module can be extended by vendors to define
  vendor-specific parameters and policies.

  Copyright (c) 2015 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC 6338; see
  the RFC itself for full legal notices.
";

revision 2017-07-01 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6833";
}

identity ms {
  base lisp:lisp-role;
  description
    "LISP Map-Server.";
}

grouping ms-counters {
  description "Group that defines map-server counters.";
  container counters {
    config false;
    description "Container for the counters";

    leaf map-registers-in {
      type yang:counter32;
      description "Number of incoming Map-Register messages";
    }

    leaf map-registers-in-auth-failed {
      type yang:counter32;
      description
        "Number of incoming Map-Register messages failed

```

```

        authentication";
    }

    leaf map-notify-records-out {
        type yang:counter32;
        description
            "Number of outgoing Map-Notify records";
    }

    leaf proxy-reply-records-out {
        type yang:counter32;
        description
            "Number of outgoing proxy Map-Reply records";
    }

    leaf map-requests-forwarded-out {
        type yang:counter32;
        description
            "Number of outgoing Map-Requests forwarded to ETR";
    }
}

augment "/lisp:lisp/lisp:lisp-router-instances"
+ "/lisp:lisp-router-instance" {
    when "lisp:lisp-role/lisp:lisp-role-type = 'lisp-ms:ms'" {
        description
            "Augment is valid when LISP device type is Map-Server.";
    }
    description
        "This augments LISP devices list with Map-Server specific
        parameters.";
    container map-server {
        presence "LISP Map-Server operation enabled";
        description
            "Map-Server parameters.";
        container sites{
            description
                "Sites to accept registrations from.";
            list site {
                key site-id;
                description
                    "Site that can send registrations.";
                leaf site-id {
                    type uint64;
                    description "Site ID";
                }
                container auth-key {

```

```
        description
        "Site authentication key.";
        leaf auth-key-value {
            type string;
            description
            "Clear text authentication key";
        }
        leaf-list auth-key-type {
            type lisp:auth-key-type;
            description
            "Authentication key type.";
        }
    }
}
container virtual-network-ids {
    description
    "Sites for which the Map-Server accepts registrations.";
    list virtual-network-identifier {
        key "vni";
        description
        "Virtual network instances in the Map-Server.";
        leaf vni {
            type lcaf:instance-id-type;
            description
            "Virtual network identifier.";
        }
    }
    container mappings {
        description
        "EIDs registered by device.";
        list mapping {
            key "eid-id";
            description
            "List of EIDs registered by device.";
            leaf eid-id {
                type lisp:eid-id;
                description
                "Id of the EID registered.";
            }
        }
        container eid-address {
            uses lcaf:lisp-address;
            description
            "EID in generic LISP address format registered
            with the Map-Server.";
        }
        leaf-list site-id {
            type uint64;
            description "Site ID";
        }
    }
}
```

```
    }
    leaf more-specifics-accepted {
        type boolean;
        default "false";
        description
            "Flag indicating if more specific prefixes
             can be registered.";
    }
    leaf mapping-expiration-timeout {
        type int16;
        units "seconds";
        default "180"; //3 times the mapregister int
        description
            "Time before mapping is expired if no new
             registrations are received.";
    }
    container mapping-records {
        description
            "Datastore of registred mappings.";
        list mapping-record{
            key xtr-id;
            description
                "Registered mapping.";
            leaf xtr-id {
                type lisp:xtr-id-type;
                description "xTR ID";
            }
            leaf site-id {
                type uint64;
                description "Site ID";
            }
            uses lisp:mapping;
        }
    }
    }
    uses ms-counters;
}

leaf mapping-system-type {
    type lisp:mapping-system-ref;
    description
        "A reference to the mapping system";
}

container summary {
    config false;
    description "Summary state information";
}
```

```

    leaf number-configured-sites {
        type uint32;
        description "Number of configured LISP sites";
    }
    leaf number-registered-sites {
        type uint32;
        description "Number of registered LISP sites";
    }
    container af-datum {
        description "Number of configured EIDs per each AF";

        list af-data {
            key "address-type";
            description "Number of configured EIDs for this AF";
            leaf address-type {
                type lcaf:lisp-address-family-ref;
                description "AF type";
            }
            leaf number-configured-eids {
                type uint32;
                description "Number of configured EIDs for this AF";
            }
            leaf number-registered-eids {
                type uint32;
                description "Number of registered EIDs for this AF";
            }
        }
    }
    uses ms-counters;
}
}
<CODE ENDS>

```

6. LISP-Map-Resolver Module

This module captures the configuration data model of a LISP Map Resolver [RFC6833]. The model also captures some operational data elements.

6.1. Module Structure

```

module: ietf-lisp-mapresolver
augment /lisp:lisp/lisp:lisp-router-instances/lisp:lisp-router-instance:
  +--rw map-resolver!
    +--rw mapping-system-type?    lisp:mapping-system-ref
    +--rw ms-address?             inet:ip-address

```

6.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp-mapresolver@2017-07-01.yang"
module ietf-lisp-mapresolver {
  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-mapresolver";
  prefix lisp-mr;
  import ietf-lisp {
    prefix lisp;
  }
  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "lisp@ietf.org";
  description
    "This YANG module defines the generic parameters for a LISP
    Map-Resolver. The module can be extended by vendors to define
    vendor-specific parameters and policies.

    Copyright (c) 2015 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD License
    set forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (http://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC 6338; see
    the RFC itself for full legal notices.
    ";
  revision 2017-07-01 {
    description
      "Initial revision.";
    reference
      "https://tools.ietf.org/html/rfc6833";
  }
  identity mr {
    base lisp:lisp-role;
    description
      "LISP Map-Resolver.";
  }
  augment "/lisp:lisp/lisp:lisp-router-instances"
  +"/lisp:lisp-router-instance" {
```

```

    when "lisp:lisp-role/lisp:lisp-role-type = 'lisp-mr:mr'" {
      description
        "Augment is valid when LISP device type is Map-Resolver.";
    }
  description
    "This augments LISP devices list with Map-Resolver specific
    parameters.";
  container map-resolver {
    presence "LISP Map-Resolver operation enabled";
    description
      "Map-Resolver parameters.";
    leaf mapping-system-type {
      type lisp:mapping-system-ref;
      description
        "A reference to the mapping system";
    }
    leaf ms-address {
      when "../mapping-system-type='lisp-mr:single-node-mapping-system'";
      type inet:ip-address;
      description
        "address to reach the Map Server when "
        + "lisp-mr:single-node-mapping-system is being used.";
    }
  }
}
}
<CODE ENDS>

```

7. LISP-Address-Types Module

This module captures the various LISP address types, and is an essential building block used in other LISP modules.

7.1. Module Definition

```

<CODE BEGINS> file "ietf-lisp-address-types@2015-11-05.yang"
module ietf-lisp-address-types {
  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types";
  prefix laddr;
  import ietf-inet-types {
    prefix inet;
    revision-date 2013-07-15;
  }
  import ietf-yang-types {
    prefix yang;
    revision-date 2013-07-15;
  }
  organization

```

```
"IETF LISP (Locator/ID Separation Protocol) Working Group";
contact
  "lisp@ietf.org";
description
  "This YANG module defines the LISP Canonical Address Formats
  (LCAF) for LISP. The module can be extended by vendors to
  define vendor-specific parameters.

  Copyright (c) 2014 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC 6338; see
  the RFC itself for full legal notices.

";
revision 2015-11-05 {
  description
    "Initial revision.";
  reference
    "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10";
}
identity lisp-address-family {
  description
    "Base identity from which identities describing LISP address
    families are derived.";
}
identity no-address-afi {
  base lisp-address-family;
  description
    "IANA Reserved.";
}
identity ipv4-afi {
  base lisp-address-family;
  description
    "IANA IPv4 address family.";
}
identity ipv4-prefix-afi {
  base lisp-address-family;
  description
    "IANA IPv4 address family prefix.";
}
```



```
identity ipv6-afi {
  base lisp-address-family;
  description
    "IANA IPv6 address family.";
}
identity ipv6-prefix-afi {
  base lisp-address-family;
  description
    "IANA IPv6 address family prefix.";
}
identity mac-afi {
  base lisp-address-family;
  description
    "IANA MAC address family.";
}
identity distinguished-name-afi {
  base lisp-address-family;
  description
    "IANA Distinguished Name address family.";
}
identity as-number-afi {
  base lisp-address-family;
  description
    "IANA AS Number address family.";
}
identity lcaf {
  base lisp-address-family;
  description
    "IANA LISP Canonical Address Format address family.";
}
identity null-address-lcaf {
  base lcaf;
  description
    "Null body LCAF type.";
}
identity afi-list-lcaf {
  base lcaf;
  description
    "AFI-List LCAF type.";
}
identity instance-id-lcaf {
  base lcaf;
  description
    "Instance-ID LCAF type.";
}
identity as-number-lcaf {
  base lcaf;
  description
```

```
        "AS Number LCAF type.";
    }
    identity application-data-lcaf {
        base lcaf;
        description
            "Application Data LCAF type.";
    }
    identity geo-coordinates-lcaf {
        base lcaf;
        description
            "Geo-coordinates LCAF type.";
    }
    identity opaque-key-lcaf {
        base lcaf;
        description
            "Opaque Key LCAF type.";
    }
    identity nat-traversal-lcaf {
        base lcaf;
        description
            "NAT-Traversal LCAF type.";
    }
    identity nonce-locator-lcaf {
        base lcaf;
        description
            "Nonce-Locator LCAF type.";
    }
    identity multicast-info-lcaf {
        base lcaf;
        description
            "Multicast Info LCAF type.";
    }
    identity explicit-locator-path-lcaf {
        base lcaf;
        description
            "Explicit Locator Path LCAF type.";
    }
    identity security-key-lcaf {
        base lcaf;
        description
            "Security Key LCAF type.";
    }
    identity source-dest-key-lcaf {
        base lcaf;
        description
            "Source/Dest LCAF type.";
    }
    identity replication-list-lcaf {
```

```
    base lcaf;
    description
      "Replication-List LCAF type.";
  }
  identity json-data-model-lcaf {
    base lcaf;
    description
      "JSON Data Model LCAF type.";
  }
  identity key-value-address-lcaf {
    base lcaf;
    description
      "Key/Value Address LCAF type.";
  }
  identity encapsulation-format-lcaf {
    base lcaf;
    description
      "Encapsulation Format LCAF type.";
  }
  identity service-path-lcaf {
    base lcaf;
    description
      "Service Path LCAF type.";
  }
  typedef instance-id-type {
    type uint32 {
      range "0..16777215";
    }
    description
      "Defines the range of values for an Instance ID.";
  }
  typedef service-path-id-type {
    type uint32 {
      range "0..16777215";
    }
    description
      "Defines the range of values for a Service Path ID.";
  }
  typedef distinguished-name-type {
    type string;
    description
      "Distinguished Name address.";
    reference
      "http://www.iana.org/assignments/address-family-numbers/
      address-family-numbers.xhtml";
  }
  typedef simple-address {
    type union {
```

```
        type inet:ip-address;
        type inet:ip-prefix;
        type yang:mac-address;
        type distinguished-name-type;
        type inet:as-number;
    }
    description
        "Union of address types that can be part of LCAFs.";
}

typedef lisp-address-family-ref {
    type identityref {
        base lisp-address-family;
    }
    description
        "LISP address family reference.";
}

typedef lcac-ref {
    type identityref {
        base lcac;
    }
    description
        "LCAF types reference.";
}

grouping lisp-address {
    description
        "Generic LISP address.";
    leaf address-type {
        type lisp-address-family-ref;
        mandatory true;
        description
            "Type of the LISP address.";
    }
    leaf virtual-network-id {
        type instance-id-type;
        description
            "Virtual Network Identifier (instance-id) of the address.";
    }
    choice address {
        description
            "Various LISP address types, including IP, MAC, and LCAF.";

        leaf no-address {
            when "../address-type = 'laddr:no-addr-afi'" {
                description
                    "When AFI is 0.";
            }
        }
    }
}
```

```
    type empty;
    description
        "No address.";
}
leaf ipv4 {
    when "../address-type = 'laddr:ipv4-afi'" {
        description
            "When AFI is IPv4.";
    }
    type inet:ipv4-address;
    description
        "IPv4 address.";
}
leaf ipv4-prefix {
    when "../address-type = 'laddr:ipv4-prefix-afi'" {
        description
            "When AFI is IPv4.";
    }
    type inet:ipv4-prefix;
    description
        "IPv4 prefix.";
}
leaf ipv6 {
    when "../address-type = 'laddr:ipv6-afi'" {
        description
            "When AFI is IPv6.";
    }
    type inet:ipv6-address;
    description
        "IPv6 address.";
}
leaf ipv6-prefix {
    when "../address-type = 'laddr:ipv6-prefix-afi'" {
        description
            "When AFI is IPv6.";
    }
    type inet:ipv6-prefix;
    description
        "IPv6 address.";
}
leaf mac {
    when "../address-type = 'laddr:mac-afi'" {
        description
            "When AFI is MAC.";
    }
    type yang:mac-address;
    description
        "MAC address.";
```

```
}
leaf distinguished-name {
  when "../address-type = 'laddr:distinguished-name-afi'" {
    description
      "When AFI is distinguished-name.";
  }
  type distinguished-name-type;
  description
    "Distinguished Name address.";
}
leaf as-number {
  when "../address-type = 'laddr:as-number-afi'" {
    description
      "When AFI is as-number.";
  }
  type inet:as-number;
  description
    "AS Number.";
}
container null-address {
  when "../address-type = 'laddr:null-address-lcaf'" {
    description
      "When LCAF type is null.";
  }
  description
    "Null body LCAF type";
  leaf address {
    type empty;
    description
      "AFI address.";
  }
}
container afi-list {
  when "../address-type = 'laddr:afi-list-lcaf'" {
    description
      "When LCAF type is AFI-List.";
  }
  description
    "AFI-List LCAF type.";
  reference
    "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10#section-4.16.1";
  leaf-list address-list {
    type simple-address;
    description
      "List of AFI addresses.";
  }
}
```

```
container instance-id {
  when "../address-type = 'laddr:instance-id-lcaf'" {
    description
      "When LCAF type is Instance-ID";
  }
  description
    "Instance ID LCAF type.";
  reference
    "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
    #section-4.2";
  leaf iid {
    type instance-id-type;
    description
      "Instance ID value.";
  }
  leaf mask-length {
    type uint8;
    description
      "Mask length.";
  }
  leaf address {
    type simple-address;
    description
      "AFI address.";
  }
}
container as-number-lcaf {
  when "../address-type = 'laddr:as-number-lcaf'" {
    description
      "When LCAF type is AS-Number.";
  }
  description
    "AS Number LCAF type.";
  reference
    "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
    #section-4.3";
  leaf as {
    type inet:as-number;
    description
      "AS number.";
  }
  leaf address {
    type simple-address;
    description
      "AFI address.";
  }
}
container application-data {
```

```
when "../address-type = 'laddr:application-data-lcaf'" {
  description
    "When LCAF type is Application Data.";
}
description
  "Application Data LCAF type.";
reference
  "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
  #section-4.4";
leaf address {
  type simple-address;
  description
    "AFI address.";
}
leaf protocol {
  type uint8;
  description
    "Protocol number.";
}
leaf ip-tos {
  type int32;
  description
    "Type of service field.";
}
leaf local-port-low {
  type inet:port-number;
  description
    "Low end of local port range.";
}
leaf local-port-high {
  type inet:port-number;
  description
    "High end of local port range.";
}
leaf remote-port-low {
  type inet:port-number;
  description
    "Low end of remote port range.";
}
leaf remote-port-high {
  type inet:port-number;
  description
    "High end of remote port range.";
}
}
container geo-coordinates {
  when "../address-type = 'laddr:geo-coordinates-lcaf'" {
    description
```



```
        "When LCAF type is Geo-coordinates.";
    }
    description
        "Geo-coordinates LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
        #section-4.5";
    leaf latitude {
        type bits {
            bit N {
                description
                    "Latitude bit.";
            }
        }
        description
            "Bit that selects between North and South latitude.";
    }
    leaf latitude-degrees {
        type uint8 {
            range "0 .. 90";
        }
        description
            "Degrees of latitude.";
    }
    leaf latitude-minutes {
        type uint8 {
            range "0..59";
        }
        description
            "Minutes of latitude.";
    }
    leaf latitude-seconds {
        type uint8 {
            range "0..59";
        }
        description
            "Seconds of latitude.";
    }
    leaf longitude {
        type bits {
            bit E {
                description
                    "Longitude bit.";
            }
        }
        description
            "Bit that selects between East and West longitude.";
    }
}
```

```
leaf longitude-degrees {
  type uint16 {
    range "0 .. 180";
  }
  description
    "Degrees of longitude.";
}
leaf longitude-minutes {
  type uint8 {
    range "0..59";
  }
  description
    "Minutes of longitude.";
}
leaf longitude-seconds {
  type uint8 {
    range "0..59";
  }
  description
    "Seconds of longitude.";
}
leaf altitude {
  type int32;
  description
    "Height relative to sea level in meters.";
}
leaf address {
  type simple-address;
  description
    "AFI address.";
}
}
container nat-traversal {
  when "../address-type = 'laddr:nat-traversal-lcaf'" {
    description
      "When LCAF type is NAT-Traversal.";
  }
  description
    "NAT-Traversal LCAF type.";
  reference
    "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
    #section-4.6";
  leaf ms-udp-port {
    type uint16;
    description
      "Map-Server UDP port (set to 4342).";
  }
  leaf etr-udp-port {
```

```
        type uint16;
        description
            "ETR UDP port.";
    }
    leaf global-etr-rloc {
        type simple-address;
        description
            "Global ETR RLOC address.";
    }
    leaf ms-rloc {
        type simple-address;
        description
            "Map-Server RLOC address.";
    }
    leaf private-etr-rloc {
        type simple-address;
        description
            "Private ETR RLOC address.";
    }
    leaf-list rtr-rlocs {
        type simple-address;
        description
            "List of RTR RLOC addresses.";
    }
}
container explicit-locator-path {
    when "../address-type = 'laddr:explicit-locator-path-lcaf'" {
        description
            "When LCAF type type is Explicit Locator Path.";
    }
    description
        "Explicit Locator Path LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
        #section-4.9";
    list hop {
        key "hop-id";
        ordered-by user;
        description
            "List of locator hops forming the explicit path.";
        leaf hop-id {
            type string {
                length "1..64";
            }
            description
                "Unique identifier for the hop.";
        }
        leaf address {
```

```
        type simple-address;
        description
            "AFI address.";
    }
    leaf lrs-bits {
        type bits{
            bit lookup {
                description
                    "Lookup bit.";
            }
            bit rloc-probe {
                description
                    "RLOC-probe bit.";
            }
            bit strict {
                description
                    "Strict bit.";
            }
        }
        description
            "Flag bits per hop.";
    }
}

container source-dest-key {
    when "../address-type = 'laddr:source-dest-key-lcaf'" {
        description
            "When LCAF type type is Source/Dest.";
    }
    description
        "Source/Dest LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
        #section-4.11";
    leaf source {
        type simple-address;
        description
            "Source address.";
    }
    leaf dest {
        type simple-address;
        description
            "Destination address.";
    }
}

container key-value-address {
    when "../address-type = 'laddr:key-value-address-lcaf'" {
        description
```

```
        "When LCAF type type is Key/Value Address.";
    }
    description
        "Key/Value Address LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
        #section-4.11";
    leaf key {
        type simple-address;
        description
            "Address as Key.";
    }
    leaf value {
        type simple-address;
        description
            "Address as Value.";
    }
}
container service-path {
    when "../address-type = 'laddr:service-path-lcaf'" {
        description
            "When LCAF type service path identifier.";
    }
    description
        "Service Path LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ermagan-lisp-nsh-00";
    leaf service-path-id {
        type service-path-id-type;
        description
            "Service path identifier for the path for NSH header";
    }
    leaf service-index {
        type uint8;
        description
            "Service path index for NSH header";
    }
}
}
```

<CODE ENDS>

8. Acknowledgments

The tree view and the YANG model shown in this document have been formatted with the 'pyang' tool.

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

Security Considerations TBD

11. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<http://www.rfc-editor.org/info/rfc6830>>.
- [RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <<http://www.rfc-editor.org/info/rfc6832>>.
- [RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, DOI 10.17487/RFC6833, January 2013, <<http://www.rfc-editor.org/info/rfc6833>>.
- [RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, DOI 10.17487/RFC6836, January 2013, <<http://www.rfc-editor.org/info/rfc6836>>.
- [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<http://www.rfc-editor.org/info/rfc8060>>.

[RFC8111] Fuller, V., Lewis, D., Ermagan, V., Jain, A., and A. Smirnov, "Locator/ID Separation Protocol Delegated Database Tree (LISP-DDT)", RFC 8111, DOI 10.17487/RFC8111, May 2017, <<http://www.rfc-editor.org/info/rfc8111>>.

Authors' Addresses

Vina Ermagan
Cisco Systems
San Jose, CA
USA

Email: vermagan@cisco.com

Alberto Rodriguez-Natal
Cisco Systems
San Jose, CA
USA

Email: natal@cisco.com

Florin Coras
Cisco Systems
San Jose, CA
USA

Email: fcoras@cisco.com

Carl Moberg
Cisco Systems
San Jose, CA
USA

Email: camoberg@cisco.com

Reshad Rahman
Cisco Systems
Canada

Email: rrahman@cisco.com

Albert Cabellos-Aparicio
Technical University of Catalonia
Barcelona
Spain

Email: acabello@ac.upc.edu

Fabio Maino
Cisco Systems
San Jose, CA
USA

Email: fmaino@cisco.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: March 29, 2018

C. Cassar
J. Leong
D. Lewis
Cisco Systems
I. Kouvelas
Arista Networks Inc.
J. Arango
Microsoft
September 25, 2017

LISP Map Server Reliable Transport
draft-kouvelas-lisp-map-server-reliable-transport-04.txt

Abstract

The communication between LISP ETRs and Map-Servers is based on unreliable UDP message exchange coupled with periodic message transmission in order to maintain soft state. The drawback of periodic messaging is the constant load imposed on both the ETR and the Map-Server. New use cases for LISP have increased the amount of state that needs to be communicated with requirements that are not satisfied by the current mechanism. This document introduces the use of a reliable transport for ETR to Map-Server communication in order to eliminate the periodic messaging overhead, while providing reliability, flow-control and endpoint liveness detection.

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 29, 2018.

Copyright Notice

Copyright (c) 2017 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
2. Requirements Notation	3
3. Message Format	4
4. Session Establishment	5
5. Error Notifications	5
6. EID Prefix Registration	7
6.1. Reliable Mapping Registration Messages	7
6.1.1. Registration Message	8
6.1.2. Registration Acknowledgement Message	8
6.1.3. Registration Rejection Message	9
6.1.4. Registration Refresh Message	10
6.1.5. Mapping Notification Message	12
6.2. ETR Behavior	13
6.3. Map-Server Behavior	17
7. Security Considerations	18
8. IANA Considerations	18
8.1. LISP Reliable Transport Message Types	18
8.2. Transport Protocol Port Numbers	18
9. Acknowledgments	18
10. Normative References	19
Authors' Addresses	19

1. Introduction

The communication channel between LISP ETRs and Map-Servers is based on unreliable UDP message exchange [RFC6833]. Where required, reliability is pursued through periodic retransmissions that maintain soft state on the peer. Map-Register messages are retransmitted every minute by an ETR and the Map-Server times out its state if the state is not refreshed for three successive periods. When registering multiple EID-Prefixes, the ETR includes multiple mapping

records in the Map-Register message. Packet size limitations provide an upper bound to the number of mapping records that can be placed in each Map-Register message. When the ETR has more EID-Prefixes to register than can be packed in a single Map-Register message, the mapping records for the EID-Prefixes are split across multiple Map-Register messages.

The drawback of the periodic registration is the constant load that it introduces on both the ETR and the Map-Server. The ETR uses resources to periodically build and transmit the Map-Register messages, and to process the resulting Map-Notify messages issued by the Map-Server. The Map-Server uses resources to process the received Map-Register messages, update the corresponding registration state, and build and transmit the matching Map-Notify messages. When the number of EID-Prefixes to be registered by an ETR is small, the resulting load imposed by periodic registrations may not be significant. The ETR will only transmit a single Map-Register message each period that contains a small number of mapping records.

In some LISP deployments, a large set of EID-Prefixes must be registered by each ETR (e.g. mobility, database redistribution). Use cases with a large set of EID-Prefixes behind an ETR will result in a much higher load. An example is LISP mobility deployments where EID-Prefixes are limited to host entries. ETRs may have thousands of hosts to register resulting in hundreds of Map-Register and Map-Notify messages per registration period.

A transport is required for the ETR to Map-Server communication that provides reliability, flow-control and endpoint liveness notifications. This document describes the use of TCP or SCTP as a LISP reliable transport. The initial application for the LISP reliable transport session is the support of scalable EID prefix registration. The reliable session mechanism is defined to be extensible so that it can support additional LISP communication requirements as they arise using a single reliable transport session between an ETR and a Map-Server. The use of the reliable transport session for EID prefix registration is an alternative and does not replace the existing UDP based mechanism.

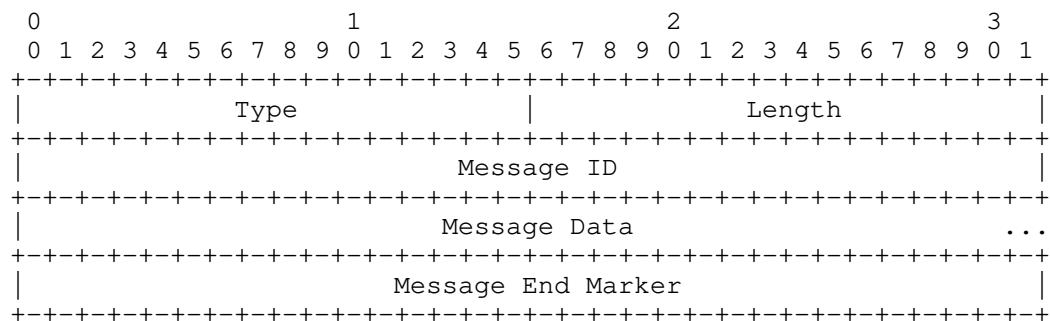
2. Requirements Notation

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

3. Message Format

A single LIISP reliable transport session may carry information for multiple LIISP applications. One such application is the registration of EID to RLOC mappings that operates over a session between an ETR and a Map-Server. Communication over a session is based on the exchange of messages. This document defines a base set of messages to support session establishment and management. It also defines the messages for the EID to RLOC mapping registration application.

To support protocol extensibility when new applications, or extensions to existing applications are introduced, the messages are based on a TLV format.



Reliable transport message format

- o Type: 16 bit type field identifying the message type.
- o Length: 16 bit field that provides the total size of the message in octets including the length, type and end marker fields. The length allows the receiver to locate the next message in the TCP stream. The minimum value of the length field is 8.
- o ID: A 32-bit value that identifies the message. May be used by the receiver to identify the message in replies or notification messages.
- o Data: Type specific message contents.
- o End Marker: A 32-bit message end marker that must be set to 0x9FACADE9. The End Marker is used by the receiver to validate that it has correctly parsed or skipped a message and provides a method to detect formatting errors. Note that message data may also contain this marker, and that the marker itself is not sufficient for parsing the message.

The base message format does not indicate how the peer should deal with the message in cases where the message type is not supported/understood. This is best dealt with by the application. For example, in case an error notification is returned, or an expected acknowledgement message is not received, the application might choose various courses of action; from simply logging that the feature is not supported, all the way to tearing the relationship with the peer down for the feature, or for all LISP features.

4. Session Establishment

To ensure backwards compatibility, the map server and ETR MUST communicate via unreliable UDP messages until a TCP session between the two is successfully established.

The map server authenticates the ETR with the authentication data contained in the first UDP map-register message it receives from the ETR. Once the ETR is authenticated, the map server performs a passive open by listening on TCP port 4342, and does not qualify the remote port. As a security measure, the map server accepts TCP connections only from those ETRs that have been authenticated via UDP map-register messages.

The ETR assumes the active role of the TCP session establishment by connecting to the map server once it has received a UDP map-notify message.

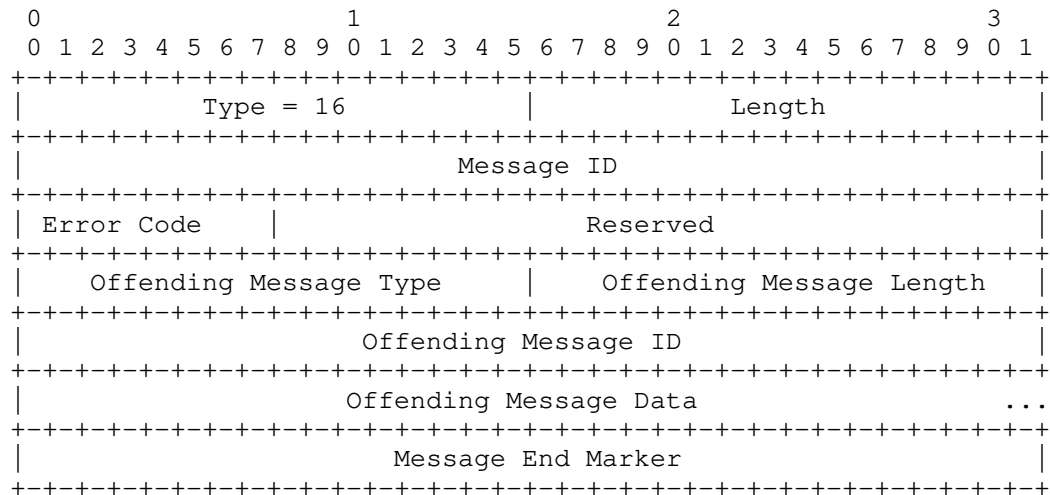
When a TCP session goes down, UDP authentication must take place before a new TCP session is established. The map-server will not accept a connection from the ETR until a UDP map-register has been received. Similarly, the ETR will not attempt to establish a session with the map server until an UDP map-notify message has been received.

A single reliable transport session is established between the map server and the ETR to cover all communication needs. For example, an ETR that has EID prefix registrations for multiple EID instances and EID address families will only establish a single session with the map server.

5. Error Notifications

The error notification message is used to communicate base reliable transport session communication errors. LISP applications making use of the reliable transport session and having to communicate application specific errors must define their own messages to do so. An error notification is issued when the receiver of a message does not recognize the message type or cannot parse the message contents.

The notification includes the offending message type and ID and as much of the offending message data as the notification sender wishes to.



Error Notification message format

- o Error Code: An 8 bit field identifying the type of error that occurred. Defined errors are:
 - * Unrecognized message type.
 - * Message format error.
- o Reserved: Set to zero by the sender and ignored by the receiver.
- o Offending Message Type: 16 bit type field identifying the message type of the offending message that triggered this error notification. This is copied from the Type field of the offending message.
- o Offending Message Length: 16 bit field that provides the total size of the offending message in octets. This is copied from the Length field of the offending message.
- o Offending Message ID: A 32-bit field that is set to the Message ID field of the offending message.
- o Offending Message Data: The Data from the offending message that triggered this error notification. The sender of the notification may include as much of the original data as is deemed necessary.

The length of the Offending Message Data field is not provided by the Offending Message Length field and is determined by subtracting the size of the other fields in the message from the Length field. It is valid to not include any of the offending message data when sending an error notification.

- o End Marker: A 32-bit message end marker that must be set to 0x9FACADE9. The End Marker is used by the receiver to validate that it has correctly parsed or skipped a message and provides a method to detect formatting errors. Note that message data may also contain this marker, and that the marker itself is not sufficient for parsing the message.

An error notification cannot be the offending message in another error notification and MUST NOT trigger such a message.

6. EID Prefix Registration

EID prefix registration uses the reliable transport session between an ETR and a Map-Server to communicate the ETR local EID database EID to RLOC mappings to the Map-Server. In contrast to the UDP based periodic registration, mapping information over the reliable transport session is only sent when there is new information available for the Map-Server. The Map-Server does not maintain a timer to expire registrations communicated over the reliable transport session. Instead an explicit de-registration (a registration carrying a zero TTL) is needed to delete the state maintained by the Map-Server.

The key used to identify registration mapping records in the ETR to Map-Server communication is the EID prefix. The prefix may be specified using an LCAF encoding that includes an EID instance ID.

When the reliable transport session goes down, registration mappings learned by the Map-Server are treated as periodic UDP registrations and a timer is used to expire them after 3 minutes. During this period UDP based registrations or the re-establishment of the reliable transport session and subsequent communication of a new mapping can update the EID prefix mapping state.

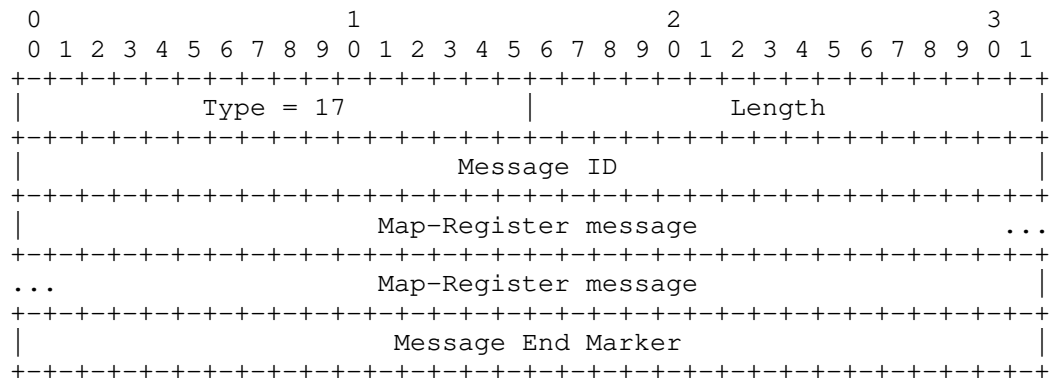
6.1. Reliable Mapping Registration Messages

This section defines the LISP reliable transport session messages used to communicate local EID database registrations between the ETR and the Map-Server.

6.1.1.1. Registration Message

The reliable transport registration message is used to communicate EID to RLOC mapping registrations from the ETR to the Map-Server. To increase code reuse, the "Message Data" field uses the same format as the UDP Map-Registers but without the IP and UDP headers. A reliable registration message MUST contain a single mapping-record. The map server MUST discard any reliable registration message that contains more than one mapping record.

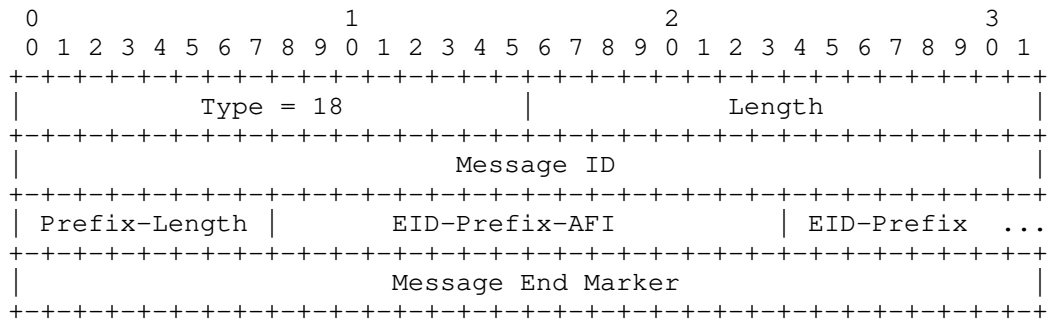
The reliable transport session is authenticated by means of the session establishment procedure. Thus, although the Map-Register MUST carry the authentication data, it is up to the map server to determine if each individual reliable registration message should be authenticated.



Registration message format

6.1.1.2. Registration Acknowledgement Message

The Acknowledgement message is sent from the Map-Server to the ETR to confirm successful registration of an EID prefix previously communicated by a reliable transport session Registration message. The Registration Acknowledgement message does not carry a mapping record (the map servers view of the mapping). This is accomplished by the LIISP reliable transport Map Notification message.

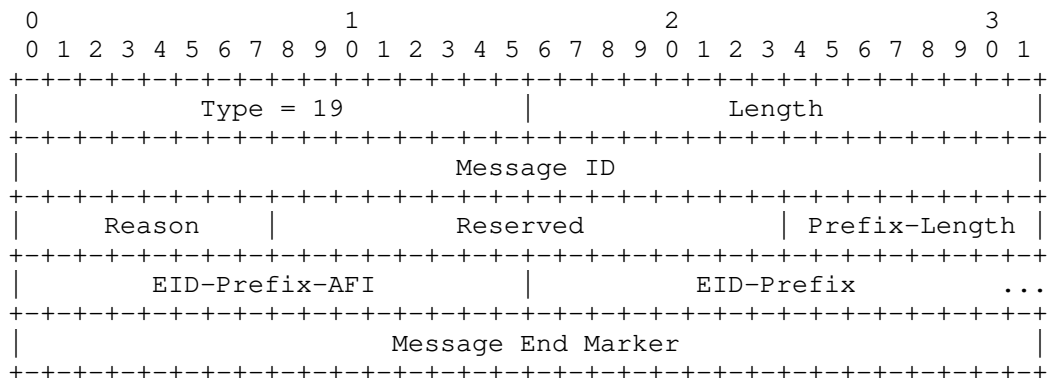


Registration Acknowledgement message format

- o Prefix-Length: Mask length for the EID prefix.
- o EID-Prefix AFI: Address family identifier for the EID prefix in the following field.
- o EID-Prefix: The EID prefix from the received Registration.

6.1.3. Registration Rejection Message

The Registration Rejection Message is sent by the map server to the ETR to indicate that the registration of a specific EID prefix is being rejected or withdrawn. A rejection refers to a recently-sent registration that is being immediately rejected. A withdrawal refers to a previously accepted registration that is no longer acceptable, perhaps due to a configuration change in the map-server. The ETR must keep track of rejected EID prefixes and be prepared to re-register their mappings when requested through a registration refresh message.

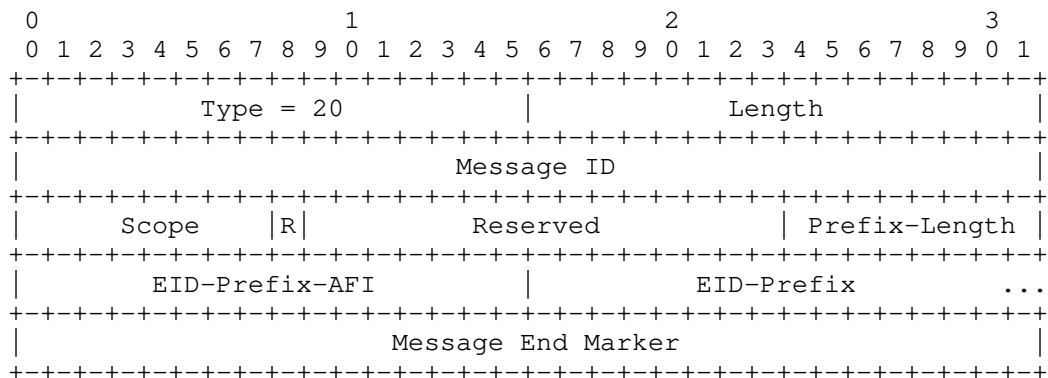


Registration Rejection message format

- o Reason: Code identifying the reason for which the Map-Server rejected or withdrew the registration.
 - * 1 - Not a valid site EID prefix.
 - * 2 - Authentication failure.
 - * 3 - Locator set not allowed.
 - * 4 - Used to cover reason that's not defined.
- o Reserved: This field is reserved for future use. Set to zero by the sender and ignored by the receiver.
- o Prefix-Length: Mask length for the EID prefix.
- o EID-Prefix-AFI: Address family identifier for the EID prefix in the following field.
- o EID-Prefix: The EID prefix being rejected or withdrawn.

6.1.4. Registration Refresh Message

Sent by the Map-Server to the ETR to request the (re-)transmission of EID prefix database mapping Registration messages.



Registration Refresh message format

- o Scope: Determines the set of registrations being refreshed.
 - * 0 - All prefixes under all address families under all EID instances are being refreshed. When using this scope the Prefix-Length, EID-Prefix-AFI, and EID-Prefix fields MUST be omitted. That is, the Message End Marker follows immediately

after the Reserved field. The total length of the message MUST be 15 bytes.

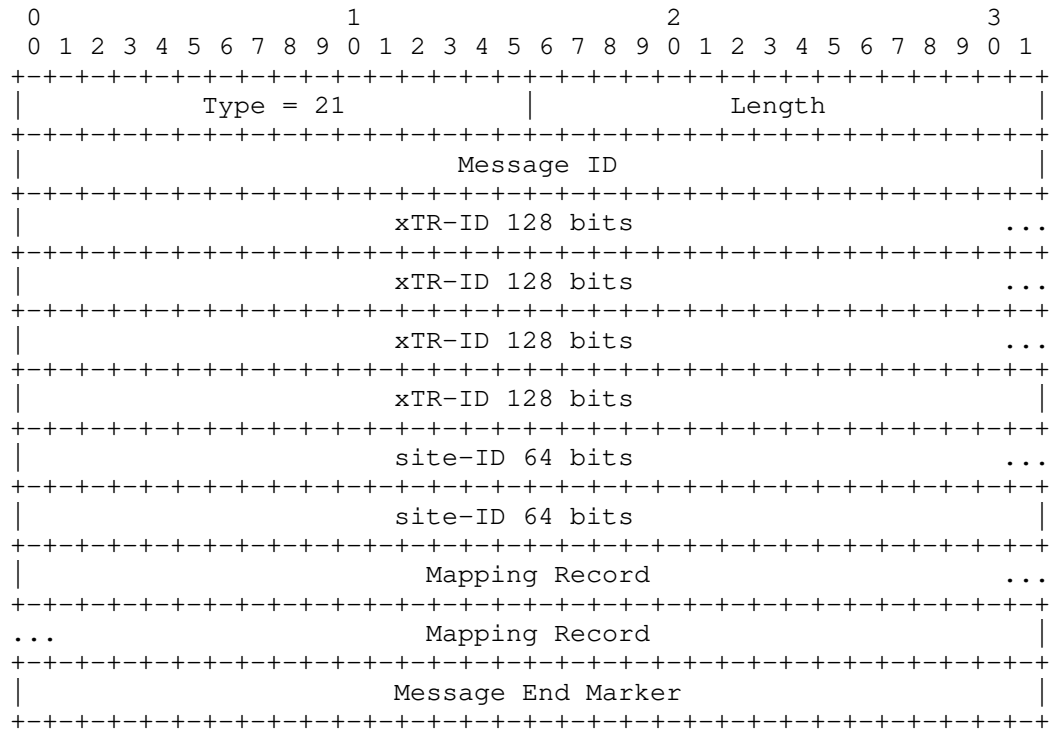
- * 1 - All prefixes under all address families under a single EID instance are being refreshed. The Prefix-Length MUST be set to zero, EID-Prefix-AFI MUST be set to LCAF type, the EID-Prefix encodes the LCAF Instance ID, the LCAF address AFI MUST be set to UNSPECIFIED. The total length of the message MUST be 30 bytes.
- * 2 - All prefixes under a single address family under a single EID instance are being refreshed. The Prefix-Length MUST be set to zero, the EID-Prefix-AFI MUST be set to LCAF type and the EID-Prefix MUST encode the Instance ID. The LCAF address AFI MUST specify the address family to refresh, the actual address SHOULD be set to zero.
- * 3 - All prefixes covered by a specific EID prefix in a single EID instance is being refreshed. The Prefix-Length, EID-Prefix-AFI and EID prefix MUST be encoded accordingly.
- * 4 - A specific EID prefix in a single EID instance is being refreshed. The Prefix-Length, EID-Prefix-AFI and EID prefix MUST be encoded accordingly.

The map-server has the flexibility to control the granularity of the refresh by issuing refresh with different scopes. It can send a single refresh with a coarse scope or send individual refreshes with narrower scope. The ETR MUST be able to process all scopes to ensure the map-server registration states are synchronized with the ETR.

- o R: Request from the ETR to only refresh registrations that have been previously rejected by the Map-Server. If the R bit is set then the scope cannot have a value of 3 and the EID-Prefix and Prefix-Length fields must be omitted.
- o Reserved: This field is reserved for future use. Set to zero by the sender and ignored by the receiver.
- o Prefix-Length: Mask length for the EID prefix. Refer to scope for more details.
- o EID-Prefix-AFI: Address family identifier for the EID prefix in the following field. Refer to scope for more details.
- o EID-Prefix: The EID prefix being refreshed. Refer to scope for more details.

6.1.5. Mapping Notification Message

Mapping Notification messages communicate the Map-Server view of the mapping for an EID prefix and no longer serve as a registration acknowledgement. Mapping Notifications do not need message level authentication as they are received over a reliable transport session to a known Map-Server. Note that reliable transport Mapping Notification messages do not reuse the UDP Map-Notify message format.



Mapping Notification message format

- o xTR-ID: xTR-ID taken from the last valid registration the map-server received for the EID-prefix conveyed in the mapping record.
- o site-ID: site-ID taken from the last valid registration the map-server received for the EID-prefix conveyed in the mapping record.
- o Mapping Record: Mapping record of the EID-prefix the map-server is conveying to the ETR.

6.2. ETR Behavior

The ETR operates the following per EID prefix, per MS state machine that defines the reliable transport EID prefix registration behavior.

There are five states:

- o No state: The local EID database prefix does not exist.
- o Periodic: The local EID database prefix is being periodically registered through UDP Map-Register messages as specified in [].
- o Stable: From the ETR's perspective, no registrations are due to be sent to the peer. The session to the peer is up, and the peer has either acknowledged the registration, or is expected to request a refresh in the future.
- o AckWait: A Registration message for the prefix has been transmitted to the Map-Server and the ETR is waiting for either a Registration Acknowledge or Registration Rejected reply from the Map-Server.
- o Reject: The reliable transport registration for the local EID database prefix was rejected by the Map-Server. From the ETR's perspective, no registration is due to the peer AND the peer is known to have rejected the registration.

The following events drive the state transitions:

- o DB creation: The local EID database entry for the EID prefix is created.
- o DB deletion: The local EID database entry for the EID prefix is deleted.
- o DB change: The mapping contents or authentication information for the local EID database entry changes.
- o Session up: The reliable transport session to the Map-Server is established.
- o Session down: The reliable transport session the Map-Server goes down.
- o Recv Refresh: A Registration refresh message is received from the Map-Server.

- o Recv ACK: A Registration Acknowledge message is received from the Map-Server.
- o Recv Rejected: A Registration Rejected message is received from the Map-Server.
- o Periodic timer: The timer that drives generation of periodic UDP Map-Register messages fires.

The state machine is:

Event	Prev State	
	No state	Periodic
DB creation [session down]	-> Periodic A1	N/A
DB creation [session up]	-> AckWait A2	N/A
DB deletion	N/A	-> No state A3
DB change	N/A	- A1
Session up	-	-> Stable A4
Session down	-	N/A
Recv Refresh	-	N/A
Recv Refresh [rejected]	-	N/A
Recv ACK	-	N/A
Recv Rejection	-	N/A
Timer	N/A	- A5

xTR per EID prefix per MS state machine

Event	Prev State		
	Stable	AckWait	Rejected
DB creation	N/A	N/A	N/A
DB deletion	-> No state A6	-> No state A6	-> No state
DB change	-> AckWait A2	- A2	-> AckWait A2
Session up	N/A	N/A	N/A
Session down	-> Periodic A7	-> Periodic A7	-> Periodic A7
Recv Refresh	-> AckWait A2	- A2	-> AckWait A2
Recv Refresh [rejected]	-	- A2	-> AckWait A2
Recv ACK	-	-> Stable	-> AckWait A2
Recv Rejection	-> Rejected	-> Rejected	-
Timer	N/A	N/A	N/A

xTR per EID prefix per MS state machine

Action descriptions:

- o A1: Start periodic registration timer with zero delay.
- o A2: Send Registration over reliable transport session.
- o A3: Send UDP registration with zero TTL.
- o A4: Stop periodic registration timer.

- o A7: Send UDP registration and start periodic registration timer with registration period.
- o A6: Send Registration with TTL zero over reliable transport session.
- o A7: Start periodic registration timer with registration period.

All timer start actions must be jittered.

When the reliable transport session is established the ETR moves the state machine into the Stable state without first registering the EID prefix over the reliable transport session. The map server will send a refresh message with a scope of 0 that will trigger the registration message to be sent. Because other applications may be using the reliable session, the refresh message signals the ETR that the map server supports reliable map registration messages. This model will also allow future optimizations where the Map-Server may retain registration state from a previous instantiation of the reliable transport session with the ETR and only request the refresh of EID prefix state beyond some negotiated session progress marker.

Aa Map-Server authentication key change is treated as a DB change event and will result in triggering a new Registration message to be transmitted.

6.3. Map-Server Behavior

Received registrations create/update or delete mapping state.

A refresh with global scope is sent when a session between the ETR and map-server is first established so the map-server can obtain the complete database contents from the ETR. This refresh is also serving as a capability signaling from the map-server to the ETR that it can support reliable registration.

Refresh for rejected registrations sent (R bit set) when a new EID prefix is configured on the Map-Server.

Refresh is sent whenever authentication key is changed or EID prefix is deconfigured. Upon reception of the registration map-server can decide whether to acknowledge the registration or issue rejection.

Mapping Notification message sent whenever the mapping for a registered or more specific prefix for which notifications are requested changes. ETR acknowledgement or rejection messaging for Mapping Notification is not required because the ETR decides how to process the message based on the registered mapping information. If

the mapping information changes the resulting registration will trigger a new Mapping Notification message from the Map-Server.

7. Security Considerations

The LISP reliable transport session SHOULD be authenticated. On controlled RLOC networks that can guarantee that the source RLOC address of data packets cannot be spoofed, the authentication check can be a source address validation on the reliable transport packets. When the RLOC network does not provide such guarantees, reliable transport authentication SHOULD be used. Implementations SHOULD support the TCP Authentication Option (TCP-AO) [RFC5925] and SCTP Authenticated Chunks [RFC4895].

8. IANA Considerations

8.1. LISP Reliable Transport Message Types

Assignment of new LISP reliable transport message types is done according to the "IETF Review" model defined in [RFC5266].

The initial content of the registry should be as follows.

Type	Name	Reference
-----	-----	-----
0-15	Reserved	This document
16	Error Notification	This document
17	Registration Message	This document
18	Registration Acknowledgement Message	This document
19	Registration Rejected Message	This document
20	Registration Refresh Message	This document
21	Mapping Notification Message	This document
22-30	Reserved for EID membership distribution	TBD
31-64999	Unassigned	
65000-65535	Reserved for Experimental Use	

8.2. Transport Protocol Port Numbers

TCP port 4342 already reserved for LISP CONS that is now obsolete. Repurpose for reliable transport over TCP. Reserve an SCTP port.

9. Acknowledgments

The authors would like to thank Noel Chiappa, Dino Farinacci, Jesper Skriver, Andre Pelletier and Les Ginsberg for their contributions to this document.

10. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5266] Devarapalli, V. and P. Eronen, "Secure Connectivity and Mobility Using Mobile IPv4 and IKEv2 Mobility and Multihoming (MOBIKE)", BCP 136, RFC 5266, DOI 10.17487/RFC5266, June 2008, <<https://www.rfc-editor.org/info/rfc5266>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, DOI 10.17487/RFC6833, January 2013, <<https://www.rfc-editor.org/info/rfc6833>>.

Authors' Addresses

Chris Cassar
Cisco Systems
10 New Square Park
Bedfont Lakes, Feltham TW14 8HA
United Kingdom

Email: ccassar@cisco.com

Johnson Leong
Cisco Systems
Tasman Drive
San Jose, CA 95134
USA

Email: joleong@cisco.com

Darrel Lewis
Cisco Systems
Tasman Drive
San Jose, CA 95134
USA

Email: darlewis@cisco.com

Isidor Kouvelas
Arista Networks Inc.
5453 Great America Parkway
Santa Clara, CA 95054
USA

Email: isidor@kouvelas.net

Jesus Arango
Microsoft

Email: jearango@microsoft.com

LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: April 21, 2018

A. Rodriguez-Natal
V. Ermagan
J. Leong
F. Maino
Cisco Systems
A. Cabellos-Aparicio
Technical University of Catalonia
S. Barkai
Fermi Serverless
D. Farinacci
lispers.net
M. Boucadair
C. Jacquenet
Orange
S. Secci
LIP6 UPMC
October 18, 2017

Publish/Subscribe Functionality for LISP
draft-rodrigueznatal-lisp-pubsub-01

Abstract

This document specifies an extension to the use of Map-Request to enable Publish/Subscribe (PubSub) operation for LISP.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 21, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Requirements Language	3
3. Deployment Assumptions	3
4. Map-Request Additions	4
5. Mapping Request Subscribe Procedures	5
6. Mapping Notification Publish Procedures	7
7. Security Considerations	8
8. Acknowledgments	8
9. IANA Considerations	8
10. Normative References	8
Authors' Addresses	8

1. Introduction

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

ITRs/RTRs/PITRs pull EID-to-RLOC mapping information from the Mapping System by means of an explicit request message. [RFC6830] indicates how ETRs can tell ITRs/RTRs/PITRs about mapping changes. This document presents a Publish/Subscribe (PubSub) extension in which the Mapping System can notify ITRs/RTRs/PITRs about mapping changes. When this mechanism is used, mapping changes can be notified faster and can be managed in the Mapping System versus the LISP sites.

In general, when an ITR/RTR/PITR wants to be notified for mapping changes for a given EID-prefix, the following steps occur:

- (1) The ITR/RTR/PITR sends a Map-Request for that EID-prefix.
- (2) The ITR/RTR/PITR sets the Notification-Requested bit (N-bit) on the Map-Request and includes its xTR-ID.
- (3) The Map-Request is forwarded to one of the Map-Servers that the EID-prefix is registered to.
- (4) The Map-Server creates subscription state for the ITR/RTR/PITR on the EID-prefix.
- (5) The Map-Server sends a Map-Notify to the ITR/RTR/PITR to acknowledge the successful subscription.
- (6) When there is an RLOC-set change for the EID-prefix, the Map-Server sends a Map-Notify message to each ITR/RTR/PITR in the subscription list.
- (7) Each ITR/RTR/PITR sends a Map-Notify-Ack to acknowledge the received Map-Notify.

This operation is repeated for all EID-prefixes for which ITR/RTR/PITR want to be notified. The ITR/RTR/PITR can set the N-bit for several EID-prefixes within a single Map-Request

2. Requirements Language

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

3. Deployment Assumptions

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

- (1) A unique 128-bit xTR-ID identifier is assigned to each xTR.
- (2) Map-Servers are configured in proxy-reply mode, i.e., they are solicited to generate and send Map-Reply messages for the mappings they are serving.
- (3) There can be either a soft-state or hard-state security association between the xTRs and the Map-Servers.

The distribution of xTR-IDs and the management of security associations are out of the scope of this document.

4. Map-Request Additions

Figure 1 shows the format of the updated Map-Request [I-D.ietf-lisp-rfc6833bis] to support the PubSub functionality.

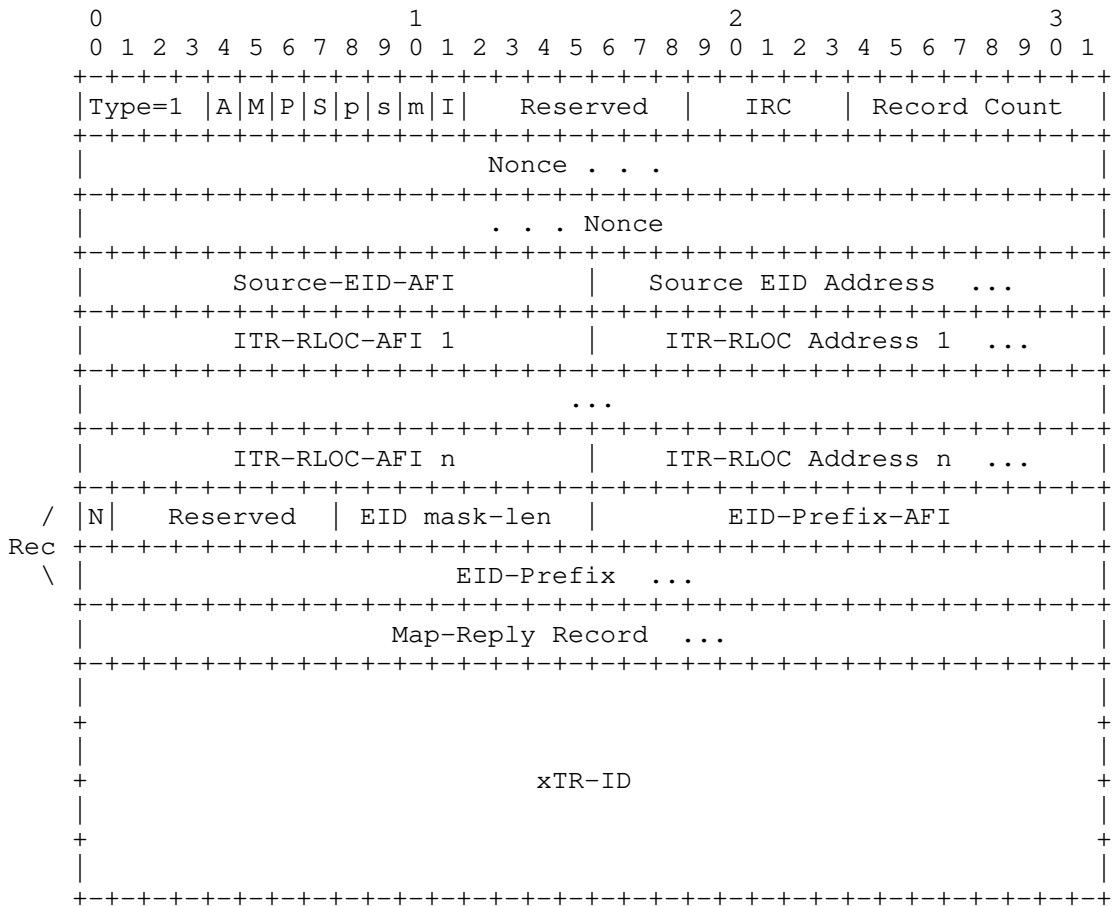


Figure 1: Map-Request with N-bit and xTR-ID

The meaning of the fields is exactly the same as defined in [I-D.ietf-lisp-rfc6833bis]. The only addition is a flag bit in the EID-Record field. The meaning of this flag bit is as follows:

Notification-Requested bit (N-bit): the first bit in the EID-Record section of a Map-Request message. The N-bit of an EID-record is set to 1 to specify that the xTR wants to be notified of updates for that mapping record.

The PubSub functionality requires to include an xTR-ID in the Map-Request. This is done by setting the xTR-ID bit (I-bit) defined in [I-D.ietf-lisp-rfc6833bis]. When the I-bit of a Map-Request message is set, a 128-bit xTR-ID field is appended to the end of the Map-Request, immediately following the last EID-Record (or the Map-Reply Record, if present). The xTR-ID field uniquely identifies each xTR of a given LIISP deployment. Provisioning of unique xTR-IDs is out of the scope of this document.

5. Mapping Request Subscribe Procedures

The xTR subscribes for RLOC-set changes for a given EID-prefix by sending a Map-Request to the Mapping System with the N-bit set on the EID-Record. The xTR builds a Map-Request according to [RFC6830] but also does the following:

- (1) The xTR MUST set the I-bit of the Map-Request message to 1, to specify the presence of an xTR-ID field that uniquely identifies the xTR.
- (2) The xTR MUST set the N-bit to 1 for each EID-Record to which the xTR wants to subscribe.

The Map-Request is forwarded to the appropriate Map-Server through the Mapping System. This document does not assume that a Map-Server is pre-assigned to handle the subscription state for a given xTR. The Map-Server that receives the Map-Request will be the Map-Server responsible to notify that specific xTR about future mapping changes for the subscribed mapping records.

Upon reception of the Map-Request, the Map-Server processes it as described in [RFC6830]. Upon processing, for each EID-Record that has the N-bit set to 1, the Map-Server proceeds adding the xTR-ID contained in the Map-Request to the list of xTR that have requested to be subscribed to that mapping record.

If the xTR-ID is added to the list, the Map-Server MUST send a Map-Notify message back to the xTR to acknowledge the successful subscription. The Map-Server MUST follow the specification in

Section 6.1.7 of [RFC6830] to build the Map-Notify with the following considerations.

- (1) The Map-Server MUST use the nonce from the Map-Request as the nonce for the Map-Notify.
- (2) The Map-Server MUST use its security association with the xTR (see Section 3) to compute the authentication data of the Map-Notify.
- (3) The Map-Server MUST send the Map-Notify to one of the ITR-RLOCs received in the Map-Request.

When the xTR receives a Map-Notify with a nonce that matches one in the list of outstanding Map-Request messages sent with an N-bit set, it knows that the Map-Notify is to acknowledge a successful subscription. The xTR processes this Map-Notify as described in [RFC6830] with the following considerations. The xTR MUST use its security association with the Map-Server (see Section 3) to validate the authentication data on the Map-Notify. The xTR MUST use the Map-Notify to populate its map-cache with the returned EID-prefix and RLOC-set.

The subscription of an xTR-ID to the list of subscribers for the EID-Record may fail for a number of reasons. For example, because of local configuration policies (such as white/black lists of subscribers), or because the Map-Server has exhausted the resources to dedicate to the subscription of that EID-Record (e.g., the number of subscribers excess the capacity of the Map-Server).

If the subscription fails, the Map-Server MUST send a Map-Reply to the originator of the Map-Request, as described in [RFC6830]. This is also the case when the Map-Server does not support PubSub operation. The xTR processes the Map-Reply as specified in [RFC6830].

If an xTR-ID is successfully added to the list of subscribers for an EID-Record, the Map-Server MUST extract the ITR-RLOCs present in the Map-Request, and store the association between the xTR-ID and those RLOCs. Any already present state regarding ITR-RLOCs for the same xTR-ID MUST be overwritten.

If the Map-Request only has one ITR-RLOC with AFI = 0 (i.e. Unknown Address), the Map-Server MUST remove the subscription state for that xTR-ID. In this case, the Map-Server MUST send the Map-Notify to the source RLOC of the Map-Request. When the TTL for the EID-record expires, the EID-prefix is removed from the Map-Server's subscription

cache. On EID-Record removal, the Map-Server notifies the subscribers via a Map-Notify with TTL equal 0.

6. Mapping Notification Publish Procedures

The publish procedure is implemented via Map-Notify messages that the Map-Server sends to xTRs. The xTRs acknowledge the reception of Map-Notifies via sending Map-Notify-Ack messages back to the Map-Server. The complete mechanism works as follows.

When a mapping stored in a Map-Server is updated (e.g. via a Map-Register from an ETR), the Map-Server MUST notify the subscribers of that mapping via sending Map-Notify messages with the most updated mapping information. The Map-Notify message sent to each of the subscribers as a result of an update event MUST follow the exact encoding and logic defined in [RFC6830] for Map-Notify, except for the following:

- (1) The Map-Notify MUST be sent to one of the ITR-RLOCs associated with the xTR-ID of the subscriber.
- (2) The nonce of the Map-Notify MUST be randomly generated by the Map-Server.
- (3) The Map-Server MUST use its security association with the xTR to compute the authentication data of the Map-Notify.

When the xTR receives a Map-Notify with a nonce not present in any list of previously sent nonces, and an EID not local to the xTR, the xTR knows that the Map-Notify has been received due to an update on the RLOC-set of a cached mapping.

The xTR processes the received Map-Notify as specified in [RFC6830], with the following considerations. The xTR MUST use its security association with the Map-Server (see Section 3) to validate the authentication data on the Map-Notify. The xTR MUST use the mapping information carried in the Map-Notify to update its internal map-cache. The xTR MUST acknowledge the Map-Notify by sending back a Map-Notify-Ack (specified in [I-D.ietf-lisp-rfc6833bis]), with the nonce from the Map-Notify, to the Map-Server. If after a configurable timeout, the Map-Server has not received back the Map-Notify-Ack, it CAN try to send the Map-Notify to a different ITR-RLOC for that xTR-ID.

7. Security Considerations

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

8. Acknowledgments

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

9. IANA Considerations

This document makes no request to IANA.

10. Normative References

- [I-D.ietf-lisp-rfc6833bis]
Fuller, V., Farinacci, D., and A. Cabellos-Aparicio,
"Locator/ID Separation Protocol (LISP) Control-Plane",
draft-ietf-lisp-rfc6833bis-06 (work in progress), October
2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The
Locator/ID Separation Protocol (LISP)", RFC 6830,
DOI 10.17487/RFC6830, January 2013,
<<https://www.rfc-editor.org/info/rfc6830>>.

Authors' Addresses

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

Email: natal@cisco.com

Vina Ermagan
Cisco Systems
170 Tasman Drive
San Jose, CA
USA

Email: vermagan@cisco.com

Johnson Leong
Cisco Systems
170 Tasman Drive
San Jose, CA
USA

Email: joleong@cisco.com

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

Email: fmaino@cisco.com

Albert Cabellos-Aparicio
Technical University of Catalonia
Barcelona
Spain

Email: acabello@ac.upc.edu

Sharon Barkai
Fermi Serverless
CA
USA

Email: sharon@fermicloud.io

Dino Farinacci
lispers.net
San Jose, CA
USA

Email: farinacci@gmail.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Christian Jacquenet
Orange
Rennes 35000
France

Email: christian.jacquenet@orange.com

Stefano Secci
LIP6 UPMC
France

Email: stefano.secci@lip6.fr