

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: July 14, 2018

D. Farinacci
lispers.net
C. Cantrell
Nexus
January 10, 2018

A Decent LISP Mapping System (LISP-Decent)
draft-farinacci-lisp-decent-00

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on July 14, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Definition of Terms	2
3. Overview	3
4. Components of a LISP-Decent xTR	5
5. No LISP Protocol Changes	6
6. Configuration and Authentication	6
7. Core Peer-Group	6
8. Security Considerations	8
9. IANA Considerations	8
10. References	8
10.1. Normative References	8
10.2. Informative References	9
Appendix A. Acknowledgments	10
Appendix B. Document Change Log	10
B.1. Changes to draft-farinacci-lisp-decent	10
Authors' Addresses	10

1. Introduction

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

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

2. Definition of Terms

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

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

Peer-Group: is a set of Map-Servers which are joined to the same multicast group that send and receive Map-Register messages addressed to the multicast group. Map-Resolvers can use the peer-group to resolve mappings by sending Map-Requests to the multicast group or to any member of the peer-group. Map-Resolvers can do a mapping system lookup for the peer-group multicast address to obtain members of the peer-group.

Core Peer-Group: is a set of Map-Servers and Map-Resolvers who are joined to a multicast group to bootstrap a multi-layer decentralized mapping system.

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

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

3. Overview

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

The xTRs are organized in a peer-group. The peer-group is identified by an IPv4 or IPv6 multicast group address. The xTRs join the same multicast group and receive LISP control-plane messages addressed to

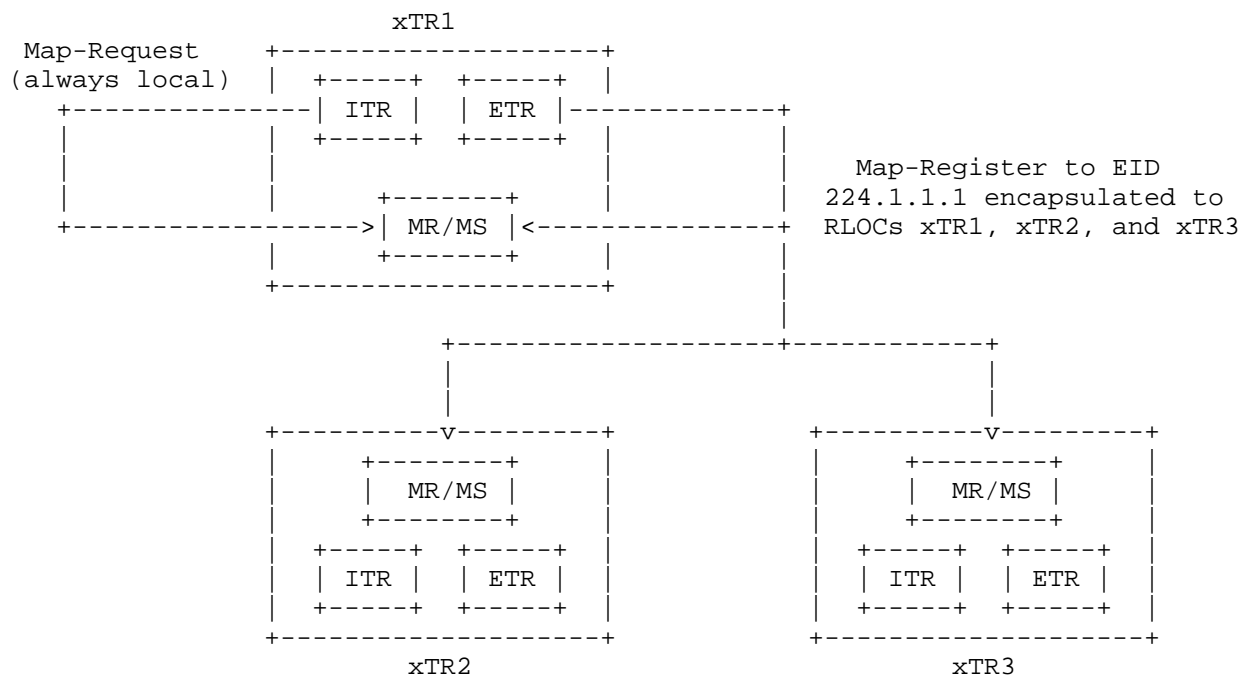
the group. Messages sent to the multicast group are distributed when the underlay network supports IP multicast [RFC6831] or is achieved with the overlay multicast mechanism described in [I-D.ietf-lisp-signal-free-multicast]. When overlay multicast is used and LISP Map-Register messages are sent to a peer-group, they are LISP data encapsulated with a instance-ID set to 0xffffffff in the LISP header. The inner header of the encapsulated packet has the destination address set to the peer-group multicast group address and the outer header that is prepended has the destination address set to the RLOC of peer-group member. The members of the peer-group are kept in the LISP data-plane map-cache so packets for the peer-group can be replicated to each member RLOC.

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

4. Components of a LISP-Decent xTR

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

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



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

For future study, an external xTR could multicast the Map-Request to 224.1.1.1 and either one of the LISP-Decent Map-Resolvers would

return a Map-Reply or the external xTR is prepared to receive multiple Map-Replies.

5. No LISP Protocol Changes

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

An ITR SHOULD lookup its peer-group address periodically to determine if the membership has changed. The ITR can also use the pubsub capability documented in [I-D.rodriqueznatal-lisp-pubsub] to be notified when a new member joins or leaves the peer-group.

6. Configuration and Authentication

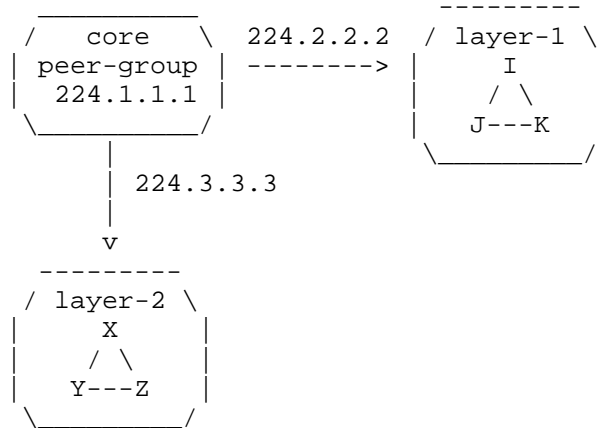
When xTRs are joined to a multicast peer-group, they must have their site registration configuration consistent. Any policy or authentication key material must be configured correctly and consistently among all members. When [I-D.farinacci-lisp-ecdsa-auth] is used to sign Map-Register messages, public-keys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

7. Core Peer-Group

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

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

This multi-layer mapping system can be illustrated:



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

224.1.1.1 -> RLE: A, B, C

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

224.2.2.2 -> RLE: I, J, K

224.3.3.3 -> RLE: X, Y, Z

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

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

...

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

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

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

...

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

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

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

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

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

8. Security Considerations

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

9. IANA Considerations

At this time there are no specific requests for IANA.

10. References

10.1. Normative References

- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC6831] Farinacci, D., Meyer, D., Zwiebel, J., and S. Venaas, "The Locator/ID Separation Protocol (LISP) for Multicast Environments", RFC 6831, DOI 10.17487/RFC6831, January 2013, <<https://www.rfc-editor.org/info/rfc6831>>.
- [RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <<https://www.rfc-editor.org/info/rfc6832>>.
- [RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, DOI 10.17487/RFC6833, January 2013, <<https://www.rfc-editor.org/info/rfc6833>>.

- [RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, DOI 10.17487/RFC6836, January 2013, <<https://www.rfc-editor.org/info/rfc6836>>.
- [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.
- [RFC8111] Fuller, V., Lewis, D., Ermagan, V., Jain, A., and A. Smirnov, "Locator/ID Separation Protocol Delegated Database Tree (LISP-DDT)", RFC 8111, DOI 10.17487/RFC8111, May 2017, <<https://www.rfc-editor.org/info/rfc8111>>.

10.2. Informative References

- [I-D.farinacci-lisp-ecdsa-auth]
Farinacci, D. and E. Nordmark, "LISP Control-Plane ECDSA Authentication and Authorization", draft-farinacci-lisp-ecdsa-auth-01 (work in progress), October 2017.
- [I-D.ietf-lisp-rfc6833bis]
Fuller, V., Farinacci, D., and A. Cabellos-Aparicio, "Locator/ID Separation Protocol (LISP) Control-Plane", draft-ietf-lisp-rfc6833bis-07 (work in progress), December 2017.
- [I-D.ietf-lisp-sec]
Maino, F., Ermagan, V., Cabellos-Aparicio, A., and D. Saucez, "LISP-Security (LISP-SEC)", draft-ietf-lisp-sec-14 (work in progress), October 2017.
- [I-D.ietf-lisp-signal-free-multicast]
Moreno, V. and D. Farinacci, "Signal-Free LISP Multicast", draft-ietf-lisp-signal-free-multicast-07 (work in progress), November 2017.
- [I-D.rodriqueznatal-lisp-pubsub]
Rodriguez-Natal, A., Ermagan, V., Leong, J., Maino, F., Cabellos-Aparicio, A., Barkai, S., Farinacci, D., Boucadair, M., Jacquenet, C., and s. stefano.secci@lip6.fr, "Publish/Subscribe Functionality for LISP", draft-rodriqueznatal-lisp-pubsub-01 (work in progress), October 2017.

Appendix A. Acknowledgments

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

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

Appendix B. Document Change Log

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

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

- o Initial draft posted January 2018.

Authors' Addresses

Dino Farinacci
lisppers.net
San Jose, CA
USA

Email: farinacci@gmail.com

Colin Cantrell
Nexus
Scottsdale, AZ
USA

Email: colin@nexus.io

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

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

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

Abstract

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

Status of This Memo

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

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

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

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

Copyright Notice

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

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

Table of Contents

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

1. Introduction

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

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

2. Definition of Terms

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

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

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

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

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

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

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

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

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

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

3. Overview

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

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

4. Push-Based Mapping System

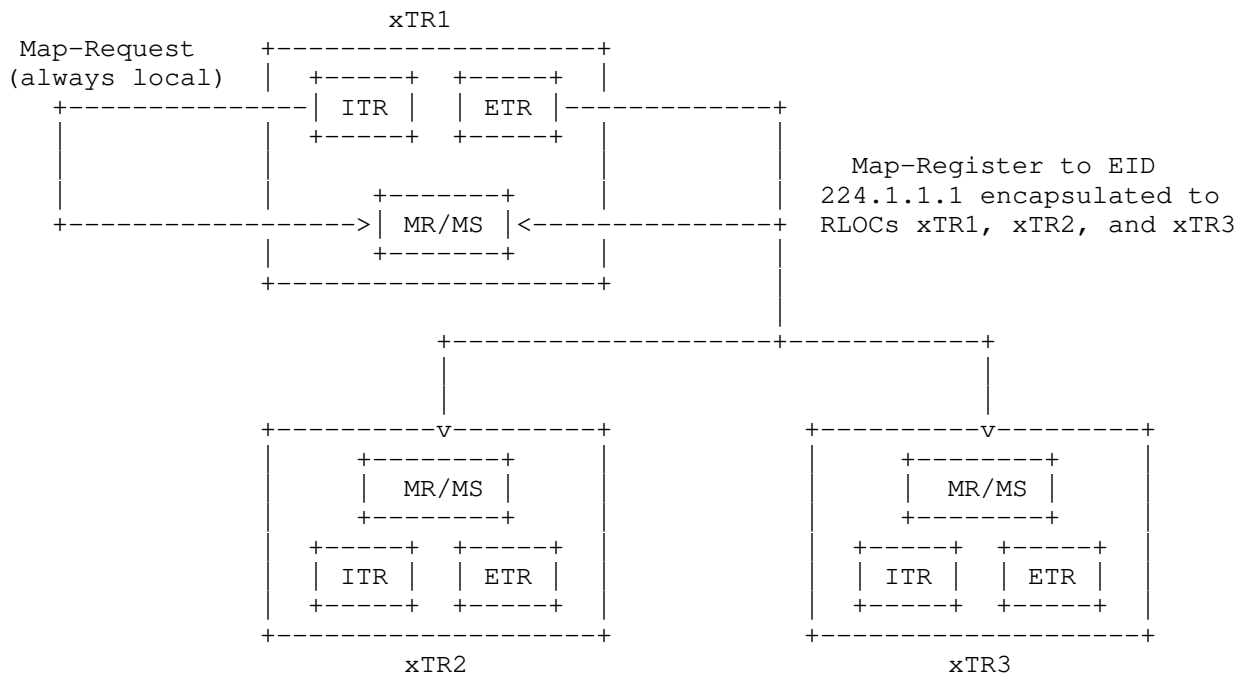
The xTRs are organized in a mapping-system group. The group is identified by an IPv4 or IPv6 multicast group address or using a pull-based approach in described in Section 5. When using multicast, the xTRs join the same multicast group and receive LISP control-plane messages addressed to the group. Messages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

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

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

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

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



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

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

4.2. No LISP Protocol Changes

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

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

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

4.3. Configuration and Authentication

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

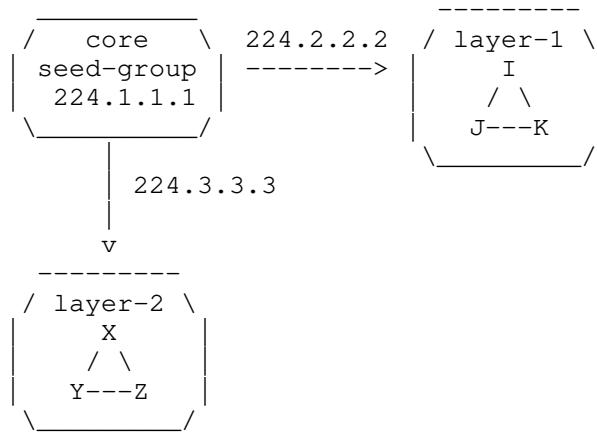
4.4. Core Seed-Group

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

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

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

This multi-layer mapping system can be illustrated:



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

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

224.2.2.2 -> RLE: I, J, K

224.3.3.3 -> RLE: X, Y, Z

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

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

...

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

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

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

...

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

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

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

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

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

5. Pull-Based Mapping System

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.2. Deployment Example

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

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

The A records for each name can be assigned as:

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

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

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

5.3. Management Considerations

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

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

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

6. Security Considerations

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

7. IANA Considerations

At this time there are no specific requests for IANA.

8. References

8.1. Normative References

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

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

8.2. Informative References

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

Appendix A. Acknowledgments

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

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

Appendix B. Document Change Log

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- o Initial draft posted January 2018.

Authors' Addresses

Dino Farinacci
lispers.net
San Jose, CA
USA

Email: farinacci@gmail.com

Colin Cantrell
Nexus
Tempe, AZ
USA

Email: colin@nexus.io

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: October 21, 2018

D. Farinacci
lispers.net
E. Nordmark
Zededa
April 19, 2018

LISP Control-Plane ECDSA Authentication and Authorization
draft-farinacci-lisp-ecdsa-auth-02

Abstract

This draft describes how LISP control-plane messages can be individually authenticated and authorized without a priori shared-key configuration. Public-key cryptography is used with no new PKI infrastructure required.

Status of This Memo

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

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

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

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

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Definition of Terms	3
3. Overview	5
4. Public-Key Hash	6
5. Hash-EID Mapping Entry	6
6. Hash-EID Structure	7
7. Keys and Signatures	7
8. Signed Map-Register Encoding	8
9. Signed Map-Request Encoding	8
10. Other Uses	9
11. EID Authorization	9
12. Security Considerations	10
13. IANA Considerations	10
14. References	10
14.1. Normative References	10
14.2. Informative References	11
Appendix A. Acknowledgments	12
Appendix B. Document Change Log	12
B.1. Changes to draft-farinacci-lisp-ecdsa-auth-02.txt	12
B.2. Changes to draft-farinacci-lisp-ecdsa-auth-01.txt	12
B.3. Changes to draft-farinacci-lisp-ecdsa-auth-00.txt	13
Authors' Addresses	13

1. Introduction

The LISP architecture and protocols [RFC6830] introduces two new numbering spaces, Endpoint Identifiers (EIDs) and Routing Locators (RLOCs) which provide an architecture to build overlays on top of the underlying Internet. Mapping EIDs to RLOC-sets is accomplished with a Mapping Database System. EIDs and RLOCs come in many forms than just IP addresses, using a general syntax that includes Address Family Identifier (AFI) [RFC1700]. Not only IP addresses, but other addressing information have privacy requirements. Access to private information is granted only to those who are authorized and authenticated. Using asymmetric keying with public key cryptography enforces authentication for entities that read from and write to the mapping system. The proposal described in this document takes advantage of the latest in Elliptic Curve Cryptography.

In this proposal the EID is derived from a public key, and the corresponding private key is used to authenticate and authorize Map-Register messages. Thus only the owner of the corresponding private key can create and update mapping entries from the EID. Furthermore, the same approach is used to authenticate Map-Request messages. This in combination with the mapping database containing authorization

information for Map-Requests is used to restrict which EIDs can lookup up the RLOCs for another EID.

This specification introduces how to use the Distinguished-Name AFI [AFI] and the [RFC8060] LCAF JSON Type to encode public keys and signatures in the LISP mapping database. The information in the mapping database is used to verify cryptographic signatures in LISP control-plane messages such as the Map-Request and Map-Register.

2. Definition of Terms

Crypto-EID: is an IPv6 EID where part of the EID includes a hash value of a public-key. An IPv6 EID is a Crypto-EID when the Map-Server is configured with an Crypto-EID Prefix that matches the IPv6 EID.

Crypto-EID Hash Length: is the number of low-order bits in a Crypto-EID which make up the hash of a public-key. The hash length is

determined by the Map-Server when it is configured with a Crypto-EID Prefix.

Crypto-EID Prefix: is a configuration parameter on the Map-Server that indicates which IPv6 EIDs are Crypto-EIDs and what is the Crypto-EID Hash Length for the IPv6 EID. This can be different for different LISP Instance-IDs.

Hash-EID: is a distinguished name EID-record stored in the mapping database. The EID format is 'hash-<pubkey-hash>'. When a key-pair is generated for an endpoint, the produced private-key does not leave the xTR that will register the Crypto-EID. A hash of the public-key is used to produce a Crypto-EID and a Hash-EID. The Crypto-EID is assigned to the endpoint and the xTR that supports the LISP-site registers the Crypto-EID. Another entity registers the Hash-EID mapping with the public-key as an RLOC-record.

Public-Key RLOC: is a JSON string that encodes a public-key as an RLOC-record for a Hash-EID mapping entry. The format of the JSON string is '{ "public-key" : "<pubkey>" }'.

Control-Plane Signature: a Map-Request or Map-Register sender signs the message with its private key. The format of the signature is a JSON string that includes sender information and the signature value. The JSON string is included in Map-Request and Map-Register messages.

Signature-EID: is a Crypto-EID used for a Control-Plane signature to register or request any type of EID. The Signature-EID is included with the JSON-encoded signature in Map-Request and Map-Register messages.

3. Overview

LISP already has several message authentication mechanisms. They can be found in [I-D.ietf-lisp-rfc6833bis], [I-D.ietf-lisp-sec], and [RFC8061]. The mechanisms in this draft are providing a more granular level of authentication as well as a simpler way to manage keys and passwords.

A client of the mapping system can be authenticated using public-key cryptography. The client is required to have a private/public key-pair where it uses the private-key to sign Map-Requests and Map-Registers. The server, or the LISP entity, that processes Map-Requests and Map-Registers uses the public-key to verify signatures.

The following describes how the mapping system is used to implement the public-key crypto system:

1. An entity registers Hash-EID to Public-Key RLOC mappings. A third-party entity that provides a service can register or the client itself can register.
2. Anyone can lookup the Hash-EID mappings. These mappings are not usually authenticated with the mechanisms in this draft but use the shared configured password mechanisms from [I-D.ietf-lisp-rfc6833bis] that provide group level authentication.
3. When a Crypto-EID, or any EID type, is registered to the mapping system, a signature is included in the Map-Register message. When a non-Crypto-EID is registered a Signature-EID is also included in the Map-Register message.
4. The Map-Server processes the registration by constructing the Hash-EID from the registered Crypto-EID, looks up the Hash-EID in the mapping system, obtains the public-key from the RLOC-record and verifies the signature. If Hash-EID lookup fails or the signature verification fails, the Map-Register is not accepted.
5. When a Crypto-EID, or any EID type, is looked up in the mapping system, a signature is included with a Signature-EID in the Map-Request message.
6. The Map-Server processes the request for a Crypto-EID by constructing the Hash-EID from the Signature-EID included in the Map-Request. The signer-EID is a Crypto-EID that accompanies a signature in the Map-Request. The Hash-EID is looked up in the mapping system, obtains the public-key from the RLOC-record and verifies the Map-Request signature. If the Hash-EID lookup fails

or the signature verification fails, the Map-Request is not accepted and a Negative Map-Reply is sent back with an action of "auth-failure".

4. Public-Key Hash

When a private/public key-pair is created for a node, its IPv6 EID is pre-determined based on the public key generated. Note if the key-pair is compromised or is changed for the node, a new IPv6 EID is assigned for the node.

The sha256 [RFC6234] hex digest function is used to compute the hash. The hash is run over the following hex byte string:

```
<iid><prefix><pubkey>
```

Where each field is defined to be:

<iid>: is a 4-byte (leading zeroes filled) binary value of the Instance-ID the EID will be registered with in the mapping database. For example, if the instance-id is 171, then the 4-byte value is 0x000000ab.

<prefix>: is a variable length IPv6 prefix in binary format (with no colons) and IS quad-nibble zero-filled. The length of the prefix is 128 minus the Crypto-EID hash bit length. For example, if the prefix is 2001:5:3::/48, then the 6 byte value is 0x200100050003.

<pubkey>: is a DER [RFC7468] encoded public-key.

The public-key hash is used to construct the Crypto-EID and Hash-EID.

5. Hash-EID Mapping Entry

A Hash-EID is formatted in an EID-record as a Distinguished-Name AFI as specified in [I-D.farinacci-lisp-name-encoding]. The format of the string is:

```
EID-record: 'hash-<hash-eid>'
```

Where <hash-eid> is a public-key hash as described in Section 4. The RLOC-record to encode and store the public-key is in LCAF JSON Type format of the form:

```
RLOC-record: '{ "public-key" : "<pubkey-base64>" }'
```

Where <pubkey-base64> is a base64 [RFC4648] encoding of the public-key generated for the system that is assigned the Hash-EID.

6. Hash-EID Structure

Since the Hash-EID is formatted as a distinguished-name AFI, the format of the <hash-eid> for EID 'hash-<hash-eid>' needs to be specified. The format will be an IPv6 address [RFC3513] where colons are used between quad-nibble characters when the hash bit length is a multiple of 4. And when the hash bit length is not a multiple of 4 but a multiple of 2, a leading 2 character nibble-pair is present. Here are some examples for different hash bit lengths:

Crypto-EID: 2001:5::1111:2222:3333:4444, hash length 64:
Hash-EID is: 'hash-1111:2222:3333:4444'

Crypto-EID: 2001:5::11:22:33:44, hash length 64:
Hash-EID is: 'hash-0011:0022:0033:0044'

Crypto-EID: 2001:5:aaaa:bbbb:1111:2222:3333:4444, hash length 80:
Hash-EID is: 'hash-bbbb:1111:2222:3333:4444'

Crypto-EID: 2001:5:aaaa:bbbb:1111:2222:3333:4444, hash length 72:
Hash-EID is: 'hash-bb:1111:2222:3333:4444'

Crypto-EID: 2001:5:aaaa:bbbb:1111:22:33:4444, hash length 72:
Hash-EID is: 'hash-bb:1111:0022:0033:4444'

Note when leading zeroes exist in a IPv6 encoded quad between colons, the zeros are included in the quad for the Hash-EID string.

The entity that creates the hash, the entity that registers the Crypto-EID and the Map-Server that uses the hash for Hash-EID lookups MUST agree on the hash bit length.

7. Keys and Signatures

Key generation, message authentication with digital signatures, and signature verification will use the Elliptic Curve Digital Signature Algorithm or ECDSA [X9.62]. For key generation curve 'NIST256p' is used and recommended.

Signatures are computed over signature data that depends on the type of LISP message sent. See Section 8 and Section 9 for each message type. The signature data is passed through a sha256 hash function before it is signed or verified.

8. Signed Map-Register Encoding

When a ETR registers its Crypto-EID or any EID type to the mapping system, it builds a LISP Map-Register message. The mapping includes an EID-record which encodes the Crypto-EID, or any EID type, and an RLOC-set. One of the RLOC-records in the RLOC-set includes the the ETR's signature and signature-EID. The RLOC-record is formatted with a LCAF JSON Type, in the following format:

```
{ "signature" : "<signature-base64>", "signature-eid" : "<signer-eid>" }
```

Where <signature-base64> is a base64 [RFC4648] encoded string over the following ascii [RFC0020] string signature data:

```
[<iid>]<crypto-eid>
```

Where <iid> is the decimal value of the instance-ID the Crypto-EID is registering to and the <crypto-eid> is in the form of [RFC3513] where quad-nibbles between colons ARE NOT zero-filled.

The Map-Server that process an EID-record with a Crypto-EID and a RLOC-record with a signature extracts the public-key hash value from the Crypto-EID to build a Hash-EID. The Map-Server looks up the Hash-EID in the mapping system to obtain the public-key RLOC-record. The Map-Server verifies the signature over the signature data to determine if it should accept the EID-record registration.

9. Signed Map-Request Encoding

When an xTR (an ITR, PITR, or RTR), sends a Map-Request to the mapping system to request the RLOC-set for a Crypto-EID, it signs the Map-Request so it can authenticate itself to the Map-Server the Crypto-EID is registered to. The Map-Request target-EID field will contain the Crypto-EID and the source-EID field will contain an LCAF JSON Type string with the following signature information:

```
{ "source-eid" : "<seid>", "signature-eid" : "<signer-eid>",  
  "signature" : "<signature-base64>" }
```

Where <signer-eid> is an IPv6 encoded string according to [RFC3513] where quad-nibbles between colons ARE NOT zero-filled. The <seid> is the source EID from the data packet that is invoking the Map-Request or the entire key/value pair for "source-eid" can be excluded when a data packet did not invoke the Map-Request (i.e. lig or an API request). The <signer-eid> is the IPv6 Crypto-EID of the xTR that is providing the Map-Request signature.

The signature string <signature-base64> is a base64 [RFC4648] encoded string over the following signature data:

<nonce><source-eid><crypto-eid>

Where <nonce> is a hex string [RFC0020] of the nonce used in the Map-Request and the <source-eid> and <crypto-eid> are hex strings [RFC0020] of an IPv6 address in the form of [RFC3513] where quad-nibbles between colons ARE NOT zero-filled. When <seid> is not included in the Map-Request, string "0::0" is used for <source-eid>.

10. Other Uses

The mechanisms described within this document can be used to sign other types of LISP messages. And for further study is how to use these mechanisms to sign LISP encapsulated data packets in a compressed manner to reduce data packet header overhead.

In addition to authenticating other types of LISP messages, other types of EID-records can be encoded as well and is not limited to IPv6 EIDs. It is possible for a LISP xTR to register and request non IPv6 EIDs but use IPv6 Crypto-EIDs for the sole purpose of signing and verifying EID-records.

Examples of other EID types that can be authenticated in Map-Request and Map-Register messages are:

EID-Type	Format Definition
IPv4 address prefixes	[RFC1123]
Distinguished-Names	[I-D.farinacci-lisp-name-encoding]
Geo-Coordinates	[I-D.farinacci-lisp-geo]
LCAF defined EIDs	[RFC8060]

11. EID Authorization

When a Crypto-EID is being used for IPv6 communication, it is implicit that the owner has the right to use the EID since it was generated from the key-pair provisioned for the owner. For other EID types that are not directly associated with signature keys, they must be validated for use by the mapping system they are registered to. This policy information for the mapping system must be configured in the Map-Servers the EID owner registers to.

12. Security Considerations

The mechanisms within this specification are intentionally using accepted practices and state of the art public-key cryptography.

Crypto-EIDs can be made private when control messages are encrypted, for instance, using [RFC8061].

The topological or physical location of a Crypto-EID is only available to the other Crypto-EIDs that register in the same LISP Instance-ID and have their corresponding Hash-EIDs registered.

This draft doesn't address reply attacks directly. If a man-in-the-middle captures Map-Register messages, it could send such captured packets at a later time which contains signatures of the source. In which case, the Map-Server verifies the signature as good and interprets the contents to be valid where in fact the contents can contain old mapping information. This problem can be solved by encrypting the contents of Map-Registers using a third-party protocol like DTLS [RFC6347] or LISP-Crypto [RFC8061] directly by encapsulating Map-Registers in LISP data packets (using port 4341).

13. IANA Considerations

Since there are no new packet formats introduced for the functionality in this specification, there are no specific requests for IANA.

14. References

14.1. Normative References

- [RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <<https://www.rfc-editor.org/info/rfc20>>.
- [RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <<https://www.rfc-editor.org/info/rfc1123>>.
- [RFC1700] Reynolds, J. and J. Postel, "Assigned Numbers", RFC 1700, DOI 10.17487/RFC1700, October 1994, <<https://www.rfc-editor.org/info/rfc1700>>.

- [RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, DOI 10.17487/RFC3513, April 2003, <<https://www.rfc-editor.org/info/rfc3513>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [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>>.
- [RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <<https://www.rfc-editor.org/info/rfc7468>>.
- [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.
- [RFC8061] Farinacci, D. and B. Weis, "Locator/ID Separation Protocol (LISP) Data-Plane Confidentiality", RFC 8061, DOI 10.17487/RFC8061, February 2017, <<https://www.rfc-editor.org/info/rfc8061>>.

14.2. Informative References

- [AFI] IANA, "Address Family Identifier (AFIs)", ADDRESS FAMILY NUMBERS <http://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml>?, February 2007.

- [I-D.farinacci-lisp-geo]
Farinacci, D., "LISP Geo-Coordinate Use-Cases", draft-farinacci-lisp-geo-05 (work in progress), April 2018.
- [I-D.farinacci-lisp-name-encoding]
Farinacci, D., "LISP Distinguished Name Encoding", draft-farinacci-lisp-name-encoding-05 (work in progress), March 2018.
- [I-D.ietf-lisp-rfc6833bis]
Fuller, V., Farinacci, D., and A. Cabellos-Aparicio, "Locator/ID Separation Protocol (LISP) Control-Plane", draft-ietf-lisp-rfc6833bis-10 (work in progress), March 2018.
- [I-D.ietf-lisp-sec]
Maino, F., Ermagan, V., Cabellos-Aparicio, A., and D. Saucez, "LISP-Security (LISP-SEC)", draft-ietf-lisp-sec-15 (work in progress), April 2018.
- [X9.62] American National Standards Institute, "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", NIST ANSI X9.62-2005, November 2005.

Appendix A. Acknowledgments

A special thanks goes to Sameer Merchant for his ideas and technical contributions to the ideas in this draft.

Appendix B. Document Change Log

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

B.1. Changes to draft-farinacci-lisp-ecdsa-auth-02.txt

- o Draft posted April 2018.
- o Generalize text to allow Map-Requeusting and Map-Registering for any EID type with a proper signature-EID and signature encoded together.

B.2. Changes to draft-farinacci-lisp-ecdsa-auth-01.txt

- o Draft posted October 2017.
- o Make it more clear what values and format the EID hash is run over.

- o Update references to newer RFCs and Internet Drafts.

B.3. Changes to draft-farinacci-lisp-ecdsa-auth-00.txt

- o Initial draft posted July 2017.

Authors' Addresses

Dino Farinacci
lispers.net
San Jose, CA
USA

Email: farinacci@gmail.com

Erik Nordmark
Zededa
Santa Clara, CA
USA

Email: erik@zededa.com

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

D. Farinacci
lispers.net
E. Nordmark
Zededa
September 4, 2018

LISP Control-Plane ECDSA Authentication and Authorization
draft-farinacci-lisp-ecdsa-auth-03

Abstract

This draft describes how LISP control-plane messages can be individually authenticated and authorized without a priori shared-key configuration. Public-key cryptography is used with no new PKI infrastructure required.

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 March 8, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Definition of Terms	4
3. Overview	5
4. Public-Key Hash	6
5. Hash-EID Mapping Entry	7
6. Hash-EID Structure	7
7. Keys and Signatures	8
8. Signed Map-Register Encoding	8
9. Signed Map-Request Encoding	9
10. Signed Map-Notify Encoding	10
11. Other Uses	10
12. EID Authorization	11
13. Security Considerations	13
14. IANA Considerations	13
15. References	13
15.1. Normative References	13
15.2. Informative References	15
Appendix A. Acknowledgments	15
Appendix B. Document Change Log	16
B.1. Changes to draft-farinacci-lisp-ecdsa-auth-03.txt	16
B.2. Changes to draft-farinacci-lisp-ecdsa-auth-02.txt	16
B.3. Changes to draft-farinacci-lisp-ecdsa-auth-01.txt	16
B.4. Changes to draft-farinacci-lisp-ecdsa-auth-00.txt	16
Authors' Addresses	16

1. Introduction

The LISP architecture and protocols [RFC6830] introduces two new numbering spaces, Endpoint Identifiers (EIDs) and Routing Locators (RLOCs) which provide an architecture to build overlays on top of the underlying Internet. Mapping EIDs to RLOC-sets is accomplished with a Mapping Database System. EIDs and RLOCs come in many forms than just IP addresses, using a general syntax that includes Address Family Identifier (AFI) [RFC1700]. Not only IP addresses, but other addressing information have privacy requirements. Access to private information is granted only to those who are authorized and authenticated. Using asymmetric keying with public key cryptography enforces authentication for entities that read from and write to the mapping system. The proposal described in this document takes advantage of the latest in Elliptic Curve Cryptography.

In this proposal the EID is derived from a public key, and the corresponding private key is used to authenticate and authorize Map-Register messages. Thus only the owner of the corresponding private key can create and update mapping entries from the EID. Furthermore, the same approach is used to authenticate Map-Request messages. This

in combination with the mapping database containing authorization information for Map-Requests is used to restrict which EIDs can lookup up the RLOCs for another EID.

This specification introduces how to use the Distinguished-Name AFI [AFI] and the [RFC8060] LCAF JSON Type to encode public keys and signatures in the LISP mapping database. The information in the mapping database is used to verify cryptographic signatures in LISP control-plane messages such as the Map-Request and Map-Register.

2. Definition of Terms

Crypto-EID: is an IPv6 EID where part of the EID includes a hash value of a public-key. An IPv6 EID is a Crypto-EID when the Map-Server is configured with an Crypto-EID Prefix that matches the IPv6 EID.

Crypto-EID Hash Length: is the number of low-order bits in a Crypto-EID which make up the hash of a public-key. The hash length is determined by the Map-Server when it is configured with a Crypto-EID Prefix.

Crypto-EID Prefix: is a configuration parameter on the Map-Server that indicates which IPv6 EIDs are Crypto-EIDs and what is the Crypto-EID Hash Length for the IPv6 EID. This can be different for different LISP Instance-IDs.

Hash-EID: is a distinguished name EID-record stored in the mapping database. The EID format is 'hash-<pubkey-hash>'. When a key-pair is generated for an endpoint, the produced private-key does not leave the xTR that will register the Crypto-EID. A hash of the public-key is used to produce a Crypto-EID and a Hash-EID. The Crypto-EID is assigned to the endpoint and the xTR that supports the LISP-site registers the Crypto-EID. Another entity registers the Hash-EID mapping with the public-key as an RLOC-record.

Public-Key RLOC: is a JSON string that encodes a public-key as an RLOC-record for a Hash-EID mapping entry. The format of the JSON string is '{ "public-key" : "<pubkey>" }'.

Control-Plane Signature: a Map-Request or Map-Register sender signs the message with its private key. The format of the signature is a JSON string that includes sender information and the signature value. The JSON string is included in Map-Request and Map-Register messages.

Signature-ID: is a Crypto-EID used for a Control-Plane signature to register or request any type of EID. The Signature-ID is included with the JSON-encoded signature in Map-Request and Map-Register messages.

Multi-Signatures: multiple signatures are used in LISP when an entity allows and authorized another entity to register an EID. There can be more than one authorizing entities that allow a registering entity to register an EID. The authorizing entities sign their own RLOC-records that are registered and merged into the registering entity's Hash-EID public-key mapping. And when

the registering entity registers the EID, all authorizing entity signatures must be verified by the Map-Server before the EID is accepted.

3. Overview

LISP already has several message authentication mechanisms. They can be found in [I-D.ietf-lisp-rfc6833bis], [I-D.ietf-lisp-sec], and [RFC8061]. The mechanisms in this draft are providing a more granular level of authentication as well as a simpler way to manage keys and passwords.

A client of the mapping system can be authenticated using public-key cryptography. The client is required to have a private/public key-pair where it uses the private-key to sign Map-Requests and Map-Registers. The server, or the LISP entity, that processes Map-Requests and Map-Registers uses the public-key to verify signatures.

The following describes how the mapping system is used to implement the public-key crypto system:

1. An entity registers Hash-EID to Public-Key RLOC mappings. A third-party entity that provides a service can register or the client itself can register.
 2. Anyone can lookup the Hash-EID mappings. These mappings are not usually authenticated with the mechanisms in this draft but use the shared configured password mechanisms from [I-D.ietf-lisp-rfc6833bis] that provide group level authentication.
 3. When a Crypto-EID, or any EID type, is registered to the mapping system, a signature is included in the Map-Register message. When a non-Crypto-EID is registered a Signature-ID is also included in the Map-Register message.
 4. The Map-Server processes the registration by constructing the Hash-EID from the registered Crypto-EID, looks up the Hash-EID in the mapping system, obtains the public-key from the RLOC-record and verifies the signature. If Hash-EID lookup fails or the signature verification fails, the Map-Register is not accepted.
 5. When a Crypto-EID, or any EID type, is looked up in the mapping system, a signature is included with a Signature-ID in the Map-Request message.
 6. The Map-Server processes the request for a Crypto-EID by constructing the Hash-EID from the Signature-ID included in the Map-Request. The signer-ID is a Crypto-EID that accompanies a signature in the Map-Request. The Hash-EID is looked up in the mapping system, obtains the public-key from the RLOC-record and verifies the Map-Request signature. If the Hash-EID lookup fails or the signature verification fails, the Map-Request is not accepted and a Negative Map-Reply is sent back with an action of "auth-failure".
4. Public-Key Hash

When a private/public key-pair is created for a node, its IPv6 EID is pre-determined based on the public key generated. Note if the key-pair is compromised or is changed for the node, a new IPv6 EID is assigned for the node.

The sha256 [RFC6234] hex digest function is used to compute the hash. The hash is run over the following hex byte string:

<iid><prefix><pubkey>

Where each field is defined to be:

<iid>: is a 4-byte (leading zeroes filled) binary value of the Instance-ID the EID will be registered with in the mapping database. For example, if the instance-id is 171, then the 4-byte value is 0x000000ab.

<prefix>: is a variable length IPv6 prefix in binary format (with no colons) and IS quad-nibble zero-filled. The length of the prefix is 128 minus the Crypto-EID hash bit length. For example, if the prefix is 2001:5:3::/48, then the 6 byte value is 0x200100050003.

<pubkey>: is a DER [RFC7468] encoded public-key.

The public-key hash is used to construct the Crypto-EID and Hash-EID.

5. Hash-EID Mapping Entry

A Hash-EID is formatted in an EID-record as a Distinguished-Name AFI as specified in [I-D.farinacci-lisp-name-encoding]. The format of the string is:

EID-record: 'hash-<hash-eid>'

Where <hash-eid> is a public-key hash as described in Section 4. The RLOC-record to encode and store the public-key is in LCAF JSON Type format of the form:

RLOC-record: '{ "public-key" : "<pubkey-base64>" }'

Where <pubkey-base64> is a base64 [RFC4648] encoding of the public-key generated for the system that is assigned the Hash-EID.

6. Hash-EID Structure

Since the Hash-EID is formatted as a distinguished-name AFI, the format of the <hash-eid> for EID 'hash-<hash-eid>' needs to be specified. The format will be an IPv6 address [RFC3513] where colons are used between quad-nibble characters when the hash bit length is a multiple of 4. And when the hash bit length is not a multiple of 4 but a multiple of 2, a leading 2 character nibble-pair is present. Here are some examples for different hash bit lengths:

Crypto-EID: 2001:5::1111:2222:3333:4444, hash length 64:
Hash-EID is: 'hash-1111:2222:3333:4444'

Crypto-EID: 2001:5::11:22:33:44, hash length 64:
Hash-EID is: 'hash-0011:0022:0033:0044'

Crypto-EID: 2001:5:aaaa:bbbb:1111:2222:3333:4444, hash length 80:
Hash-EID is: 'hash-bbbb:1111:2222:3333:4444'

Crypto-EID: 2001:5:aaaa:bbbb:1111:2222:3333:4444, hash length 72:
Hash-EID is: 'hash-bb:1111:2222:3333:4444'

Crypto-EID: 2001:5:aaaa:bbbb:1111:22:33:4444, hash length 72:
Hash-EID is: 'hash-bb:1111:0022:0033:4444'

Note when leading zeroes exist in a IPv6 encoded quad between colons, the zeros are included in the quad for the Hash-EID string.

The entity that creates the hash, the entity that registers the Crypto-EID and the Map-Server that uses the hash for Hash-EID lookups MUST agree on the hash bit length.

7. Keys and Signatures

Key generation, message authentication with digital signatures, and signature verification will use the Elliptic Curve Digital Signature Algorithm or ECDSA [X9.62]. For key generation curve 'NIST256p' is used and recommended.

Signatures are computed over signature data that depends on the type of LISP message sent. See Section 8 and Section 9 for each message type. The signature data is passed through a sha256 hash function before it is signed or verified.

8. Signed Map-Register Encoding

When a ETR registers its Crypto-EID or any EID type to the mapping system, it builds a LISP Map-Register message. The mapping includes an EID-record which encodes the Crypto-EID, or any EID type, and an RLOC-set. One of the RLOC-records in the RLOC-set includes the the ETR's signature and signature-ID. The RLOC-record is formatted with a LCAF JSON Type, in the following format:

```
{ "signature" : "<signature-base64>", "signature-id" : "<signer-id>" }
```

Where <signature-base64> is a base64 [RFC4648] encoded string over the following ascii [RFC0020] string signature data:

[<iid>]<crypto-eid>

Where <iid> is the decimal value of the instance-ID the Crypto-EID is registering to and the <crypto-eid> is in the form of [RFC3513] where quad-nibbles between colons ARE NOT zero-filled.

The Map-Server that process an EID-record with a Crypto-EID and a RLOC-record with a signature extracts the public-key hash value from the Crypto-EID to build a Hash-EID. The Map-Server looks up the Hash-EID in the mapping system to obtain the public-key RLOC-record. The Map-Server verifies the signature over the signature data to determine if it should accept the EID-record registration.

9. Signed Map-Request Encoding

When an xTR (an ITR, PITR, or RTR), sends a Map-Request to the mapping system to request the RLOC-set for a Crypto-EID, it signs the Map-Request so it can authenticate itself to the Map-Server the Crypto-EID is registered to. The Map-Request target-EID field will contain the Crypto-EID and the source-EID field will contain an LCAF JSON Type string with the following signature information:

```
{
  "source-eid" : "<seid>",
  "signature-id" : "<signer-id>",
  "signature" : "<signature-base64>"
}
```

Where <signer-id> is an IPv6 encoded string according to [RFC3513] where quad-nibbles between colons ARE NOT zero-filled. The <seid> is the source EID from the data packet that is invoking the Map-Request or the entire key/value pair for "source-eid" can be excluded when a data packet did not invoke the Map-Request (i.e. lig or an API request). The <signer-id> is the IPv6 Crypto-EID of the xTR that is providing the Map-Request signature.

The signature string <signature-base64> is a base64 [RFC4648] encoded string over the following signature data:

<nonce><source-eid><crypto-eid>

Where <nonce> is a hex string [RFC0020] of the nonce used in the Map-Request and the <source-eid> and <crypto-eid> are hex strings [RFC0020] of an IPv6 address in the form of [RFC3513] where quad-nibbles between colons ARE NOT zero-filled. When <seid> is not included in the Map-Request, string "0::0" is used for <source-eid>.

10. Signed Map-Notify Encoding

When a Map-Server originates a Map-Notify message either as an acknowledgment to a Map-Register message, as a solicited [I-D.ietf-lisp-pubsub] notification, or an unsolicited [RFC8378] notification, the receiver of the Map-Notify can verify the message is from an authenticated Map-Server.

An RLOC-record similar to the one used to sign Map-Register messages is used to sign the Map-Notify message:

```
{ "signature" : "<signature-base64>", "signature-id" : "<signer-id>" }
```

Where the "signature-id" is an IPv6 crypto-EID used by the Map-Server to sign the RLOC-record. The signature data and the encoding format of the signature is the same as for a Map-Register message. See details in Section 8.

A receiver of a Map-Notify message will lookup the signature-id in the mapping system to obtain a public-key to verify the signature. The Map-Notify is accepted only if the verification is successful.

11. Other Uses

The mechanisms described within this document can be used to sign other types of LISP messages. And for further study is how to use these mechanisms to sign LISP encapsulated data packets in a compressed manner to reduce data packet header overhead.

In addition to authenticating other types of LISP messages, other types of EID-records can be encoded as well and is not limited to IPv6 EIDs. It is possible for a LISP xTR to register and request non IPv6 EIDs but use IPv6 Crypto-EIDs for the sole purpose of signing and verifying EID-records.

Examples of other EID types that can be authenticated in Map-Request and Map-Register messages are:

EID-Type	Format Definition
IPv4 address prefixes	[RFC1123]
Distinguished-Names	[I-D.farinacci-lisp-name-encoding]
Geo-Coordinates	[I-D.farinacci-lisp-geo]
LCAF defined EIDs	[RFC8060]

12. EID Authorization

When a Crypto-EID is being used for IPv6 communication, it is implicit that the owner has the right to use the EID since it was generated from the key-pair provisioned for the owner. For other EID types that are not directly associated with signature keys, they must be validated for use by the mapping system they are registered to. This policy information for the mapping system must be configured in the Map-Servers the EID owner registers to or a signed authorization provided by a third-party entity.

To achieve signed authorization, an entity that allows another entity to register an EID, must authorize the registering entity. It does so by adding RLOC-records to the registering entity's Hash-EID public-key mapping. The format of the RLOC-record is a JSON encoded record as follows:

```
{
  "allow-eid" : "<eid>",
  "signature-id" : "<signer-id>",
  "signature" : "<signature-base64>"
}
```

The format of the <signer-id> and <signature-base64> values are the same as described in Section 8. The <eid> value is in the same string format as the signature data described in Section 8. For other non-IPv6 EID types, the conventions in [RFC8060] are used. In all cases, the string encoding format of instance-ID '[<iid>]' prepended is to the EID string.

This entry is added to the RLOC-set of the registering entity's Hash-EID 'hash-<hash>' registration. The authorizing entity does signs the Map-Register and sends it with merge-semantics. The Map-Server accepts the registration after the signature is verified and merges the RLOC-record into the existing RLOC-set. The 'signature' is optional and when not included means the authorizing entity has not

yet allowed the registering entity to register the EID <eid>. Note multiple entities can register RLOC-records with the same <eid> meaning that signature verification for all of them is required before the Map-Server accepts the registration.

When the Map-Server receives a Map-Register for <eid>, it looks up 'hash-<hash>' EID in the mapping system. If not found, the Map-Register EID-record is not processed and the next EID-record is retrieved from the Map-Register message, if it exists. If the Hash-EID entry is found, the registering entity's signature is verified first. If the verification fails, the Map-Register EID-record is not accepted. Otherwise, a search for the RLOC-set is done to look for all matches of the EID being registered with <eid>, for those entries found, if any of them do not have a "signature" JSON item, the EID-record is not accepted. Otherwise, the signature-id is looked up in the mapping system to retrieve the public-key of the authorizing entity. If the verification is successful, then a lookup for the next RLOC-record signature-id is done. Only when all signature verifications are verified, the Map-Register EID-record is accepted.

The Map-Server should reject an RLOC-record with a signature-id that contains the Hash-EID of the entry disallowing a registering entity to self authorize itself.

Here is an example of a Hash-EID mapping stored in the mapping system:

EID-record: [1000]'hash-1111:2222:3333:4444', RLOC-Set (count is 4):

```
RLOC-record: { "public-key" : "<pubkey-base64>" }
RLOC-record: { "allow-eid" : "[1000]1.1.1.1/32", "signature" : "<sig>",
               "signature-id" : "[1000]2001:5:3::1111" }
RLOC-record: { "allow-eid" : "[1000]1.1.1.1/32", "signature" : "<sig>",
               "signature-id" : "[1000]2001:5:3::2222" }
RLOC-record: { "allow-eid" : "37-16-46-N-121-52-4-W",
               "signature-id" : "[1000]2001:5:3::5555" }
```

This mapping stores the public-key of the registering entity with Hash-EID 1111:2222:3333:4444. The registering entry registered this RLOC-record. There are two authorizing entities, :1111 and :2222, who allow it to register IPv4 EID 1.1.1.1/32. They each registered their respective RLOC-records. And a third authorizing entity :5555 that registers an RLOC-record that has not yet authorized the registering entity to register Geo-Coordinate 37-16-46-N-121-52-4-W. Note the mapping and the signature-IDs are all within the context of instance-ID 1000.

13. Security Considerations

The mechanisms within this specification are intentionally using accepted practices and state of the art public-key cryptography.

Crypto-EIDs can be made private when control messages are encrypted, for instance, using [RFC8061].

The topological or physical location of a Crypto-EID is only available to the other Crypto-EIDs that register in the same LISP Instance-ID and have their corresponding Hash-EIDs registered.

This draft doesn't address reply attacks directly. If a man-in-the-middle captures Map-Register messages, it could send such captured packets at a later time which contains signatures of the source. In which case, the Map-Server verifies the signature as good and interprets the contents to be valid where in fact the contents can contain old mapping information. This problem can be solved by encrypting the contents of Map-Registers using a third-party protocol like DTLS [RFC6347] or LISP-Crypto [RFC8061] directly by encapsulating Map-Registers in LISP data packets (using port 4341).

Map-Reply message signatures and authentication are not in scope for this document. This document focuses on authentication between xTRs and mapping system components. Map-Reply authentication, which is performed between xTRs is described in [I-D.ietf-lisp-sec].

14. IANA Considerations

Since there are no new packet formats introduced for the functionality in this specification, there are no specific requests for IANA.

15. References

15.1. Normative References

- [RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <<https://www.rfc-editor.org/info/rfc20>>.
- [RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <<https://www.rfc-editor.org/info/rfc1123>>.

- [RFC1700] Reynolds, J. and J. Postel, "Assigned Numbers", RFC 1700, DOI 10.17487/RFC1700, October 1994, <<https://www.rfc-editor.org/info/rfc1700>>.
- [RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, DOI 10.17487/RFC3513, April 2003, <<https://www.rfc-editor.org/info/rfc3513>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [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>>.
- [RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <<https://www.rfc-editor.org/info/rfc7468>>.
- [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.
- [RFC8061] Farinacci, D. and B. Weis, "Locator/ID Separation Protocol (LISP) Data-Plane Confidentiality", RFC 8061, DOI 10.17487/RFC8061, February 2017, <<https://www.rfc-editor.org/info/rfc8061>>.

- [RFC8378] Moreno, V. and D. Farinacci, "Signal-Free Locator/ID Separation Protocol (LISP) Multicast", RFC 8378, DOI 10.17487/RFC8378, May 2018, <<https://www.rfc-editor.org/info/rfc8378>>.

15.2. Informative References

- [AFI] IANA, "Address Family Identifier (AFIs)", ADDRESS FAMILY NUMBERS <http://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml>?, February 2007.
- [I-D.farinacci-lisp-geo] Farinacci, D., "LISP Geo-Coordinate Use-Cases", draft-farinacci-lisp-geo-05 (work in progress), April 2018.
- [I-D.farinacci-lisp-name-encoding] Farinacci, D., "LISP Distinguished Name Encoding", draft-farinacci-lisp-name-encoding-05 (work in progress), March 2018.
- [I-D.ietf-lisp-pubsub] Rodriguez-Natal, A., Ermagan, V., Leong, J., Maino, F., Cabellos-Aparicio, A., Barkai, S., Farinacci, D., Boucadair, M., Jacquenet, C., and S. Secci, "Publish/Subscribe Functionality for LISP", draft-ietf-lisp-pubsub-00 (work in progress), April 2018.
- [I-D.ietf-lisp-rfc6833bis] Fuller, V., Farinacci, D., and A. Cabellos-Aparicio, "Locator/ID Separation Protocol (LISP) Control-Plane", draft-ietf-lisp-rfc6833bis-13 (work in progress), August 2018.
- [I-D.ietf-lisp-sec] Maino, F., Ermagan, V., Cabellos-Aparicio, A., and D. Saucez, "LISP-Security (LISP-SEC)", draft-ietf-lisp-sec-15 (work in progress), April 2018.
- [X9.62] American National Standards Institute, "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", NIST ANSI X9.62-2005, November 2005.

Appendix A. Acknowledgments

A special thanks goes to Sameer Merchant and Colin Cantrell for their ideas and technical contributions to the ideas in this draft.

Appendix B. Document Change Log

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

B.1. Changes to draft-farinacci-lisp-ecdsa-auth-03.txt

- o Posted September 2018.
- o Change all occurrences of signature-EID to signature-ID.
- o Document how Map-Servers sign Map-Notify messages so they can be verified by xTRs.
- o Add multi-signatures to mappings so a 3rd-party can allow an entity to register any type of EID.

B.2. Changes to draft-farinacci-lisp-ecdsa-auth-02.txt

- o Draft posted April 2018.
- o Generalize text to allow Map-Requesting and Map-Registering for any EID type with a proper signature-EID and signature encoded together.

B.3. Changes to draft-farinacci-lisp-ecdsa-auth-01.txt

- o Draft posted October 2017.
- o Make it more clear what values and format the EID hash is run over.
- o Update references to newer RFCs and Internet Drafts.

B.4. Changes to draft-farinacci-lisp-ecdsa-auth-00.txt

- o Initial draft posted July 2017.

Authors' Addresses

Dino Farinacci
lispers.net
San Jose, CA
USA

Email: farinacci@gmail.com

Erik Nordmark
Zededa
Santa Clara, CA
USA

Email: erik@zededa.com

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

D. Farinacci
lispers.net
S. Ouissal
E. Nordmark
Zededa
June 27, 2018

LISP Data-Plane Telemetry
draft-farinacci-lisp-telemetry-00

Abstract

This draft specs a JSON formatted RLOC-record for telemetry data which decapsulating xTRs include in RLOC-probe Map Reply messages.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 29, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Definition of Terms	3
3. Overview	4
4. Telemetry Record Encoding	5
5. Security Considerations	6
6. IANA Considerations	7
7. References	7
7.1. Normative References	7
7.2. Informative References	7
Appendix A. Acknowledgments	7
Appendix B. Document Change Log	8
B.1. Changes to draft-farinacci-lisp-telemetry-00	8
Authors' Addresses	8

1. Introduction

This document describes how the Locator/Identifier Separation Protocol (LISP) can obtain, measure, and distribute data-plane telemetry information. LISP is an encapsulation protocol built around the fundamental idea of separating the topological location of a network attachment point from the node's identity [I-D.ietf-lisp-rfc6830bis]. As a result LISP creates two namespaces: Endpoint Identifiers (EIDs), that are used to identify end-hosts and routable Routing Locators (RLOCs), used to identify network attachment points. LISP then defines functions for mapping between the two namespaces and for encapsulating traffic originated by devices using non-routable EIDs for transport across a network infrastructure that routes and forwards using RLOCs.

This document specifies how a decapsulating xTR returns telemetry data to an encapsulating xTR using RLOC-probe messages defined in [I-D.ietf-lisp-rfc6833bis].

Early versions of this document will define the type and format of the telemetry data and how it will be distributed. Later versions of this document will describe how telemetry measurement will be performed.

2. Definition of Terms

Encapsulating xTR is a LISP ITR, RTR, or PITR data-plane network element [I-D.ietf-lisp-rfc6830bis]. An encapsulating xTR typically sends RLOC-probe Map-Request messages to decapsulating xTRs to test for reachability of RLOC addresses. For the design scope of this specification, RLOC-probes are also sent to obtain LISP telemetry data measured by a decapsulating xTR.

Decapsulating xTR is a LISP ETR, RTR, or PETR data-plane network element [I-D.ietf-lisp-rfc6830bis]. A decapsulating xTR typically RLOC-probe replies with a Map-Reply message to an RLOC-probe Map-Request sent by an encapsulating xTR. When a decapsulating xTR does data-plane telemetry measurement, it returns measurement data in RLOC-probe Map-Reply messages to an encapsulating xTR.

Telemetry Record a telemetry record is an RLOC-record that contains telemetry data specified in this document. The telemetry data is encoded as an LCAF JSON Type specified in [RFC8060].

3. Overview

The following list of telemetry data has been identified as being useful to obtain:

- o Packet Count - the number of packets received within a given time window between the encapsulating xTR and decapsulating xTR.
- o Byte Count - the number bytes summed from all packets received within a given time window between the encapsulating xTR and decapsulating xTR.
- o Packet Rate - the rate in packets per second an encapsulating xTR is sending encapsulated packets to a decapsulating xTR.
- o Bit Rate - the bit rate per second an encapsulating xTR is sending encapsulated packets to a decapsulating xTR.
- o Bandwidth - the amount of bandwidth used between encapsulating xTR and decapsulating xTR in bytes per second.
- o Packet Loss - the number of packets lost within a given time window between the encapsulating xTR and decapsulating xTR.
- o Packet Jitter - the amount of inter-packet time for a train of packets within a given time window between the encapsulating xTR and decapsulating xTR.
- o Forward Hop-Count - the number underlay router hops from the encapsulating xTR to the decapsulating xTR.
- o Forward One-Way Latency - the amount of time from the encapsulating xTR to the decapsulating xTR. Available when a universal clock and rough time synchronization is available.
- o Reverse TTL - the TTL value a decapsulating xTR is using for the RLOC-probe Map-Reply. This is used to compute the return or Reverse Hop-Count or number of underlay router hops between the decapsulating xTR and encapsulating xTR.
- o Reverse Timestamp - the universal clock timestamp when the decapsulating xTR sent the RLOC-probe Map-Reply message. This is used to compute the return or Reverse One-Way Latency between the decapsulating xTR to the encapsulating xTR.

4. Telemetry Record Encoding

A Telemetry Record is an RLOC-record encoded in LCAF JSON Type format [RFC8060] within the EID-record inserted in a RLOC-probe Map-Reply message. The RLOC-record is appended to the existing RLOC-records for the EID being probed.

An encapsulating xTR does not need to request telemetry data so the decapsulating xTR can provide it unilaterally by default or via configuration to enable the feature. When an encapsulating xTR receives a Telemetry Record in a RLOC-probe Map-Reply, it SHOULD NOT store it in the map-cache and not use the RLOC-record for forwarding (since there are no RLOCs in this record). The priority for this RLOC-record MUST be set to 255 and the weight MUST be set to 0.

The JSON key values imply directionality. The directionality is from encapsulating xTR to decapsulating xTR. That is, the same direction of RLOC-probe Map-Requests and encapsulated packet flow. The JSON string format is defined to be:

```
{ "type" :          "telemetry",
  "packet-count" :   "<pc>",
  "packet-loss" :    "<pl>",
  "byte-count" :     "<bc>",
  "packet-rate" :    "<pr>",
  "bit-rate" :       "<br>",
  "bandwidth" :      "<b>",
  "packet-jitter" :  "<pj>",
  "forward-latency" : "<fl>",
  "forward-hop-count" : "<hc>",
  "reverse-ttl" :    "<ttn>",
  "reverse-timestamp" : "<ts>"
}
```

JSON data values:

JSON Value	Encoding Description
<pc>	Number of packets encoded as an integer value within a string.
<pl>	Number of lost packets encoded as an integer value within a string.
<bc>	Number of bytes encoded as an integer value within a string.
<pr>	Packet rate in packets per second encoded as an integer value within a string.
 	Bit rate in kilobits per second encoded as an integer value within a string.
	Bandwidth in kilobytes encoded as an integer value within a string.
<pj>	Packet jitter in milliseconds encoded as an integer value within a string.
<fl>	Latency in milliseconds encoded as an integer value within a string.
<hc>	Hop count encoded as an integer value within a string.
<tth>	Map-Reply IP header TTL encoded as an integer value within a string.
<ts>	Timestamp encoded in Linux UTC format as an within a string (i.e. Tue Jun 26 16:27:25 UTC 2018).

5. Security Considerations

RLOC-probe Map-Reply messages are signed to protect and authenticate the Telemetry Record according to details in [I-D.ietf-lisp-sec]. Telemetry Records can be kept confidential by encrypting RLOC-probe Map-Reply message with the asymmetric keys described in [I-D.farinacci-lisp-ecdsa-auth] or the symmetric keys computed by the key exchange detailed in [RFC8061].

6. IANA Considerations

At this time there are no specific requests for IANA.

7. References

7.1. Normative References

[RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.

[RFC8061] Farinacci, D. and B. Weis, "Locator/ID Separation Protocol (LISP) Data-Plane Confidentiality", RFC 8061, DOI 10.17487/RFC8061, February 2017, <<https://www.rfc-editor.org/info/rfc8061>>.

7.2. Informative References

[I-D.farinacci-lisp-ecdsa-auth]
Farinacci, D. and E. Nordmark, "LISP Control-Plane ECDSA Authentication and Authorization", draft-farinacci-lisp-ecdsa-auth-02 (work in progress), April 2018.

[I-D.ietf-lisp-rfc6830bis]
Farinacci, D., Fuller, V., Meyer, D., Lewis, D., and A. Cabellos-Aparicio, "The Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-rfc6830bis-12 (work in progress), March 2018.

[I-D.ietf-lisp-rfc6833bis]
Fuller, V., Farinacci, D., and A. Cabellos-Aparicio, "Locator/ID Separation Protocol (LISP) Control-Plane", draft-ietf-lisp-rfc6833bis-10 (work in progress), March 2018.

[I-D.ietf-lisp-sec]
Maino, F., Ermagan, V., Cabellos-Aparicio, A., and D. Saucez, "LISP-Security (LISP-SEC)", draft-ietf-lisp-sec-15 (work in progress), April 2018.

Appendix A. Acknowledgments

The authors would like to thank the LISP WG for their review and acceptance of this draft. A special thanks to Colin Cantrell for his review, commentary and guidance.

Appendix B. Document Change Log

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

B.1. Changes to draft-farinacci-lisp-telemetry-00

- o Initial draft posted June 2018.

Authors' Addresses

Dino Farinacci
lispers.net
San Jose, CA
USA

Email: farinacci@gmail.com

Said Ouissal
Zededa
Santa Clara, CA
USA

Email: said@zededa.com

Erik Nordmark
Zededa
Santa Clara, CA
USA

Email: erik@zededa.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 10 November 2022

D. Farinacci
lispers.net
S. Ouissal
E. Nordmark
Zededa
9 May 2022

LISP Data-Plane Telemetry
draft-farinacci-lisp-telemetry-08

Abstract

This draft specs a JSON formatted RLOC-record for telemetry data which decapsulating xTRs include in RLOC-probe Map Reply messages.

Status of This Memo

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

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

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

This Internet-Draft will expire on 10 November 2022.

Copyright Notice

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

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

Table of Contents

1. Introduction	2
2. Definition of Terms	3
3. Overview	3
4. Telemetry Record Encoding	4
5. Security Considerations	6
6. IANA Considerations	7
7. References	7
7.1. Normative References	7
7.2. Informative References	7
Appendix A. Acknowledgments	8
Appendix B. Document Change Log	8
B.1. Changes to draft-farinacci-lisp-telemetry-08	8
B.2. Changes to draft-farinacci-lisp-telemetry-07	8
B.3. Changes to draft-farinacci-lisp-telemetry-06	8
B.4. Changes to draft-farinacci-lisp-telemetry-05	8
B.5. Changes to draft-farinacci-lisp-telemetry-04	8
B.6. Changes to draft-farinacci-lisp-telemetry-03	8
B.7. Changes to draft-farinacci-lisp-telemetry-02	8
B.8. Changes to draft-farinacci-lisp-telemetry-01	9
B.9. Changes to draft-farinacci-lisp-telemetry-00	9
Authors' Addresses	9

1. Introduction

This document describes how the Locator/Identifier Separation Protocol (LISP) can obtain, measure, and distribute data-plane telemetry information. LISP is an encapsulation protocol built around the fundamental idea of separating the topological location of a network attachment point from the node's identity [I-D.ietf-lisp-rfc6830bis]. As a result LISP creates two namespaces: Endpoint Identifiers (EIDs), that are used to identify end-hosts and routable Routing Locators (RLOCs), used to identify network attachment points. LISP then defines functions for mapping between the two namespaces and for encapsulating traffic originated by devices using non-routable EIDs for transport across a network infrastructure that routes and forwards using RLOCs.

This document specifies how a decapsulating xTR returns telemetry data to an encapsulating xTR using RLOC-probe messages defined in [I-D.ietf-lisp-rfc6833bis].

Early versions of this document will define the type and format of the telemetry data and how it will be distributed. Later versions of this document will describe how telemetry measurement will be performed.

2. Definition of Terms

Encapsulating xTR is a LISP ITR, RTR, or PITR data-plane network element [I-D.ietf-lisp-rfc6830bis]. An encapsulating xTR typically sends RLOC-probe Map-Request messages to decapsulating xTRs to test for reachability of RLOC addresses. For the design scope of this specification, RLOC-probes are also sent to obtain LISP telemetry data measured by a decapsulating xTR.

Decapsulating xTR is a LISP ETR, RTR, or PETR data-plane network element [I-D.ietf-lisp-rfc6830bis]. A decapsulating xTR typically RLOC-probe replies with a Map-Reply message to an RLOC-probe Map-Request sent by an encapsulating xTR. When a decapsulating xTR does data-plane telemetry measurement, it returns measurement data in RLOC-probe Map-Reply messages to an encapsulating xTR.

Telemetry Record a telemetry record is an RLOC-record that contains telemetry data specified in this document. The telemetry data is encoded as an LCAF JSON Type specified in [RFC8060].

3. Overview

The following list of telemetry data has been identified as being useful to obtain:

- * Packet Count - the number of packets received within a given time window between the encapsulating xTR and decapsulating xTR.
- * Byte Count - the number bytes summed from all packets received within a given time window between the encapsulating xTR and decapsulating xTR.
- * Packet Rate - the rate in packets per second an encapsulating xTR is sending encapsulated packets to a decapsulating xTR.
- * Bit Rate - the bit rate per second an encapsulating xTR is sending encapsulated packets to a decapsulating xTR.
- * Bandwidth - the amount of bandwidth used between encapsulating xTR and decapsulating xTR in bytes per second.
- * Packet Loss - the number of packets lost within a given time window between the encapsulating xTR and decapsulating xTR.
- * Packet Jitter - the amount of inter-packet time for a train of packets within a given time window between the encapsulating xTR and decapsulating xTR.

- * Forward Hop-Count - the number underlay router hops from the encapsulating xTR to the decapsulating xTR.
- * Forward One-Way Latency - the amount of time from the encapsulating xTR to the decapsulating xTR. Available when a universal clock and rough time synchronization is available.
- * Reverse TTL - the TTL value a decapsulating xTR is using for the RLOC-probe Map-Reply. This is used to compute the return or Reverse Hop-Count or number of underlay router hops between the decapsulating xTR and encapsulating xTR.
- * Reverse Timestamp - the universal clock timestamp when the decapsulating xTR sent the RLOC-probe Map-Reply message. This is used to compute the return or Reverse One-Way Latency between the decapsulating xTR to the encapsulating xTR.

4. Telemetry Record Encoding

A Telemetry Record is an RLOC-record encoded in LCAF JSON Type format [RFC8060] within the EID-record inserted in a RLOC-probe Map-Reply message. The RLOC-record is appended to the existing RLOC-records for the EID being probed.

An encapsulating xTR does not need to request telemetry data so the decapsulating xTR can provide it unilaterally by default or via configuration to enable the feature. When an encapsulating xTR receives a Telemetry Record in a RLOC-probe Map-Reply, it SHOULD NOT store it in the map-cache and not use the RLOC-record for forwarding (since there are no RLOCs in this record). The priority for this RLOC-record MUST be set to 255 and the weight MUST be set to 0.

The JSON key values imply directionality. The directionality is from encapsulating xTR to decapsulating xTR. That is, the same direction of RLOC-probe Map-Requests and encapsulated packet flow. The JSON string format is defined to be:

```
{ "type" :          "telemetry",
  "packet-count" :   "<pc>",
  "packet-loss" :    "<pl>",
  "byte-count" :     "<bc>",
  "packet-rate" :    "<pr>",
  "bit-rate" :       "<br>",
  "bandwidth" :      "<b>",
  "packet-jitter" :  "<pj>",
  "forward-latency" : "<fl>",
  "forward-hop-count" : "<hc>",
  "reverse-ttl" :    "<ttl>",
  "reverse-timestamp" : "<ts>"
}
```

JSON data values:

JSON Value	Encoding Description
<pc>	Number of packets encoded as an integer value within a string.
<pl>	Number of lost packets encoded as an integer value within a string.
<bc>	Number of bytes encoded as an integer value within a string.
<pr>	Packet rate in packets per second encoded as an integer value within a string.
 	Bit rate in kilobits per second encoded as an integer value within a string.
	Bandwidth in kilobytes encoded as an integer value within a string.
<pj>	Packet jitter in milliseconds encoded as an integer value within a string.
<fl>	Latency in milliseconds encoded as an integer value within a string.
<hc>	Hop count encoded as an integer value within a string.
<tth>	Map-Reply IP header TTL encoded as an integer value within a string.
<ts>	Timestamp encoded in Linux UTC format as an within a string (i.e. Tue Jun 26 16:27:25 UTC 2018).

Table 1

5. Security Considerations

RLOC-probe Map-Reply messages are signed to protect and authenticate the Telemetry Record according to details in [I-D.ietf-lisp-sec]. Telemetry Records can be kept confidential by encrypting RLOC-probe Map-Reply message with the asymmetric keys described in [I-D.ietf-lisp-ecdsa-auth] or the symmetric keys computed by the key exchange detailed in [RFC8061].

6. IANA Considerations

At this time there are no specific requests for IANA.

7. References

7.1. Normative References

- [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.
- [RFC8061] Farinacci, D. and B. Weis, "Locator/ID Separation Protocol (LISP) Data-Plane Confidentiality", RFC 8061, DOI 10.17487/RFC8061, February 2017, <<https://www.rfc-editor.org/info/rfc8061>>.

7.2. Informative References

- [I-D.ietf-lisp-ecdsa-auth]
Farinacci, D. and E. Nordmark, "LISP Control-Plane ECDSA Authentication and Authorization", Work in Progress, Internet-Draft, draft-ietf-lisp-ecdsa-auth-07, 21 February 2022, <<https://www.ietf.org/archive/id/draft-ietf-lisp-ecdsa-auth-07.txt>>.
- [I-D.ietf-lisp-rfc6830bis]
Farinacci, D., Fuller, V., Meyer, D., Lewis, D., and A. Cabellos, "The Locator/ID Separation Protocol (LISP)", Work in Progress, Internet-Draft, draft-ietf-lisp-rfc6830bis-38, 7 May 2022, <<https://www.ietf.org/archive/id/draft-ietf-lisp-rfc6830bis-38.txt>>.
- [I-D.ietf-lisp-rfc6833bis]
Farinacci, D., Maino, F., Fuller, V., and A. Cabellos, "Locator/ID Separation Protocol (LISP) Control-Plane", Work in Progress, Internet-Draft, draft-ietf-lisp-rfc6833bis-31, 2 May 2022, <<https://www.ietf.org/archive/id/draft-ietf-lisp-rfc6833bis-31.txt>>.
- [I-D.ietf-lisp-sec]
Maino, F., Ermagan, V., Cabellos, A., and D. Saucez, "LISP-Security (LISP-SEC)", Work in Progress, Internet-Draft, draft-ietf-lisp-sec-25, 9 December 2021, <<https://www.ietf.org/archive/id/draft-ietf-lisp-sec-25.txt>>.

Appendix A. Acknowledgments

The authors would like to thank the LISP WG for their review and acceptance of this draft. A special thanks to Colin Cantrell for his review, commentary and guidance.

Appendix B. Document Change Log

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

B.1. Changes to draft-farinacci-lisp-telemetry-08

- * Posted May 2022.
- * Document timer and reference update.

B.2. Changes to draft-farinacci-lisp-telemetry-07

- * Posted November 2021.
- * Document timer and reference update.

B.3. Changes to draft-farinacci-lisp-telemetry-06

- * Posted May 2021.
- * Document timer and reference update.

B.4. Changes to draft-farinacci-lisp-telemetry-05

- * Posted November 2020.
- * Document timer and reference update.

B.5. Changes to draft-farinacci-lisp-telemetry-04

- * Posted June 2020.
- * Document timer and reference update.

B.6. Changes to draft-farinacci-lisp-telemetry-03

- * Posted December 2019.
- * Document timer and reference update.

B.7. Changes to draft-farinacci-lisp-telemetry-02

- * Posted June 2019.

- * Document timer and reference update.

B.8. Changes to draft-farinacci-lisp-telemetry-01

- * Posted December 2018.

- * Document timer and reference update.

B.9. Changes to draft-farinacci-lisp-telemetry-00

- * Initial draft posted June 2018.

Authors' Addresses

Dino Farinacci
lisppers.net
San Jose, CA
United States of America
Email: farinacci@gmail.com

Said Ouissal
Zededa
Santa Clara, CA
United States of America
Email: said@zededa.com

Erik Nordmark
Zededa
Santa Clara, CA
United States of America
Email: erik@zededa.com

LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: December 31, 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
June 29, 2018

LISP YANG Model
draft-ietf-lisp-yang-08

Abstract

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

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA).

Status of This Memo

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

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

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

This Internet-Draft will expire on December 31, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	2
1.1. Requirements Language	3
1.2. Tree Diagrams	3
2. LISP Module	3
2.1. Module Structure	3
2.2. Module Definition	6
3. LISP-ITR Module	16
3.1. Module Structure	16
3.2. Module Definition	21
4. LISP-ETR Module	25
4.1. Module Structure	25
4.2. Module Definition	27
5. LISP-Map-Server Module	31
5.1. Module Structure	32
5.2. Module Definition	40
6. LISP-Map-Resolver Module	46
6.1. Module Structure	46
6.2. Module Definition	47
7. LISP-Address-Types Module	49
7.1. Module Definition	49
7.2. Data Model examples	63
7.2.1. LISP protocol instance	64
7.2.2. LISP ITR	65
7.2.3. LISP ETR	65
7.2.4. LISP Map-Server	67
8. Acknowledgments	68
9. IANA Considerations	68
10. Security Considerations	70
11. Normative References	70
Authors' Addresses	71

1. Introduction

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

1.1. Requirements Language

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

1.2. Tree Diagrams

This document uses the graphical representation of data models defined in [RFC8340].

2. LISP Module

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

2.1. Module Structure

```

module: ietf-lisp
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol:
      +--rw lisp
        +--rw locator-sets
          +--rw locator-set* [locator-set-name]
            +--rw locator-set-name      string
            +--rw (locator-type)?
              +--:(local-interface)
                +--rw interface* [interface-ref]
                  +--rw interface-ref    if:interface-ref
                  +--rw priority?         uint8
                  +--rw weight?           uint8
                  +--rw multicast-priority? uint8
                  +--rw multicast-weight? uint8
              +--:(general-locator)
                +--rw locator* [id]
                  +--rw id                string
                  +--rw locator-address
                    +--rw address-type
                    |   lisp-address-family-ref
                    +--rw 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-role* [lisp-role-type]
|   +--rw lisp-role-type   lisp-role-ref
+--rw lisp-router-id
|   +--rw site-id?        uint64
|   +--rw xtr-id?         lisp:xtr-id-type
+--rw virtual-networks
|   +--rw virtual-network* [vni]
|       +--rw vni          lcaf:instance-id-type
|       +--rw ni-name?
|           -> /ni:network-instances/network-instance/name

```

2.2. Module Definition

```

<CODE BEGINS> file "ietf-lisp@2018-06-29.yang"
module ietf-lisp {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp";

  prefix lisp;

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }
  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note
  import ietf-lisp-address-types {
    prefix lcaf;
    reference "RFC XXXX: LISP YANG model";
  }
  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management
      (NMDA version)";
  }
  import ietf-network-instance {
    prefix "ni";
    // RFC Ed.: replace occurrences of YYYY with actual RFC number
    // of draft-ietf-rtgwg-ni-model and remove this note
  }

```

```
reference
  "RFC YYYY: YANG Model for Network Instances";
}

organization
  "IETF LISP (Locator/ID Separation Protocol) Working Group";
contact
  "WG Web:    <http://tools.ietf.org/wg/lisp/>
  WG List:    <mailto:lisp@ietf.org>

  Editor:     Vina Ermagan
               <mailto:vermagan@cisco.com>

  Editor:     Alberto Rodriguez-Natal
               <mailto:natal@cisco.com>

  Editor:     Reshad Rahman
               <mailto:rrahman@cisco.com>";
description
  "This YANG module defines the generic parameters for LISP.
  The module can be extended by vendors to define vendor-specific
  LISP parameters and policies.

  Copyright (c) 2018 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

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

reference "RFC XXXX";

revision 2018-06-29 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6830";
}

/*
 * Identity definitions
```

```
*/
identity lisp {
  base "rt:control-plane-protocol";
  description "LISP protocol.";
  reference
    "RFC 6830: The Locator/ID Separation Protocol (LISP).";
}

identity lisp-role {
  description
    "LISP router role.";
}
identity itr {
  base lisp-role;
  description
    "LISP ITR.";
}
identity pitr {
  base lisp-role;
  description
    "LISP PITR.";
}
identity etr {
  base lisp-role;
  description
    "LISP ETR.";
}
identity petr {
  base lisp-role;
  description
    "LISP PETR.";
}
identity mapping-system {
  description
    "Mapping System interface";
}
identity single-node-mapping-system {
  base mapping-system;
  description
    "logically singular Map Server";
}
typedef mapping-system-ref {
  type identityref {
    base mapping-system;
  }
  description
    "Mapping System reference";
}
```



```
typedef lisp-role-ref {
  type identityref {
    base lisp-role;
  }
  description
    "LISP role reference";
}
typedef map-reply-action {
  type enumeration {
    enum no-action {
      value 0;
      description
        "Mapping is kept alive and no encapsulation occurs.";
    }
    enum natively-forward {
      value 1;
      description
        "Matching packets are not encapsulated or dropped but
        natively forwarded.";
    }
    enum send-map-request {
      value 2;
      description
        "Matching packets invoke Map-Requests.";
    }
    enum drop {
      value 3;
      description
        "Matching packets are dropped.";
    }
  }
  description
    "Defines the lisp map-cache ACT type";
  reference "https://tools.ietf.org/html/rfc6830#section-6.1.4";
}
typedef eid-id {
  type string;
  description
    "Type encoding of lisp-addresses to be generally used in EID
    keyed lists.";
}
typedef auth-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;
    config false;
    description
      "Time when the mapping was created.";
  }
  leaf authoritative {
    type bits {
      bit A {
        description
          "Authoritative bit.";
      }
    }
    description
      "Bit that indicates if mapping comes from an
        authoritative source.";
  }
  leaf static {
    type boolean;
    default "false";
    description
      "This leaf should be true if the mapping is static.";
  }
  choice locator-list {
    description
      "list of locartors are either negative, or positive.";
    case negative-mapping {
      leaf map-reply-action {
        type map-reply-action;
        description
          "Forwarding action for a negative mapping.";
      }
    }
    case positive-mapping {
      container rlocs {
        uses locators-grouping;
        description
          "List of locators for a positive mapping.";
      }
    }
  }
}
```

```

    }
  }
}

grouping mappings {
  description
    "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;
      }
    }
  }
}

augment "/rt:routing/rt:control-plane-protocols"
+ "/rt:control-plane-protocol" {
  when "derived-from-or-self(rt:type, 'lisp:lisp')" {
    description
      "This augmentation is only valid for a control-plane protocol
      instance of LISP.";
  }
  description "LISP protocol ietf-routing module
  control-plane-protocol augmentation.";

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

```

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

list lisp-role {
  key lisp-role-type;
  description
    "List of lisp device roles such as MS, MR, ITR,
    PITR, ETR or PETR.";
  leaf lisp-role-type {
    type lisp-role-ref;
    description
      "The type of LISP device - identity derived from the
      'lisp-device' base identity.";
  }
}

container lisp-router-id {
```

```
when "../lisp-role/lisp-role-type = 'itr' or
    ../lisp-role/lisp-role-type = 'pitr' or
    ../lisp-role/lisp-role-type = 'etr' or
    ../lisp-role/lisp-role-type = 'petr'" {
    description "Only when ITR, PITR, ETR or PETR.";
}
description
    "Site-ID and xTR-ID of the device.";
leaf site-id {
    type uint64;
    description "Site ID";
}
leaf xtr-id {
    type lisp:xtr-id-type;
    description "xTR ID";
}
}

container virtual-networks {
    description "Virtual networks";
    list virtual-network {
        key vni;
        description "List of virtual networks";

        leaf vni {
            type lcaf:instance-id-type;
            description
                "Virtual network identifier";
        }
        leaf ni-name {
            type leafref {
                path "/ni:network-instances/ni:network-instance/ni:name";
            }
            description
                "Name of Network Instance (e.g. VRF) to which a VNI is
                bound. Each VNI is bound to a different Network
                Instance";
        }
    }
}
}
}
}
<CODE ENDS>
```

3. LISP-ITR Module

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

3.1. Module Structure

```

module: ietf-lisp-itr
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/lisp:lisp:
      +--rw itr!
        +--rw rloc-probing!
          |   +--rw interval?          uint16
          |   +--rw retries?           uint8
          |   +--rw retries-interval?  uint16
        +--rw itr-rlocs?               leafref
        +--rw map-resolvers
          |   +--rw map-resolver*      inet:ip-address
        +--rw proxy-etrs
          |   +--rw proxy-etr-address*  inet:ip-address
        +--rw map-cache
          +--ro size?                  uint32
          +--ro limit?                 uint32
          +--rw virtual-network* [vni]
            +--rw vni                  lcaf:instance-id-type
            +--rw mappings
              +--rw mapping* [id]
                +--rw id                eid-id
                +--rw eid
                  +--rw address-type
                  |   +--rw lisp-address-family-ref
                +--rw virtual-network-id?
                  |   +--rw 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?                uint32
|         +--rw address?
|             simple-address
+---:(nat-traversal)
|   +--rw nat-traversal
|       +--rw ms-udp-port?                uint16
|       +--rw etr-udp-port?              uint16
|       +--rw global-etr-rloc?
|           | simple-address
|       +--rw ms-rloc?
|           | simple-address
|       +--rw private-etr-rloc?
|           | simple-address
|       +--rw rtr-rlocs*
|           simple-address
+---:(explicit-locator-path)
|   +--rw explicit-locator-path
|       +--rw hop* [hop-id]
|           +--rw hop-id                string
|           +--rw address?              simple-address
|           +--rw lrs-bits?             bits
+---:(source-dest-key)
|   +--rw source-dest-key
|       +--rw source?                    simple-address
|       +--rw dest?                      simple-address
+---:(key-value-address)
|   +--rw key-value-address
|       +--rw key?                        simple-address
|       +--rw value?                      simple-address
+---:(service-path)
|   +--rw service-path
|       +--rw service-path-id?
|           | service-path-id-type
|       +--rw service-index?            uint8
+--rw time-to-live?                      uint32
+--ro creation-time?                      yang:date-and-time
+--rw authoritative?                      bits
+--rw static?                             boolean
+--rw (locator-list)?
+---:(negative-mapping)
|   +--rw map-reply-action?              map-reply-action
+---:(positive-mapping)
+--rw rlocs
+--rw locator* [id]
+--rw id                                  string
+--rw locator-address
|   +--rw address-type
|       | lisp-address-family-ref

```

```

|--rw 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
```

3.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp-itr@2018-06-29.yang"
module ietf-lisp-itr {
    yang-version 1.1;

    namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-itr";

    prefix lisp-itr;

    // RFC Ed.: replace occurrences of XXXX with actual RFC number
    // and remove this note
    import ietf-lisp {
        prefix lisp;
    }
}
```

```
    reference "RFC XXXX: LISP YANG model";
  }
  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management
      (NMDA version)";
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "WG Web:    <http://tools.ietf.org/wg/lisp/>
    WG List:    <mailto:lisp@ietf.org>

    Editor:     Vina Ermagan
                <mailto:vermagan@cisco.com>

    Editor:     Alberto Rodriguez-Natal
                <mailto:natal@cisco.com>

    Editor:     Reshad Rahman
                <mailto:rrahman@cisco.com>";
  description
    "This YANG module defines the generic parameters for a LISP
    ITR. The module can be extended by vendors to define
    vendor-specific parameters and policies.

    Copyright (c) 2018 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

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

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

  reference "RFC XXXX";
```

```
revision 2018-06-29 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6830";
}
augment "/rt:routing/rt:control-plane-protocols"
+ "/rt:control-plane-protocol/lisp:lisp" {
  when "lisp:lisp-role/lisp:lisp-role-type = 'lisp:itr' or
    lisp:lisp-role/lisp:lisp-role-type = 'lisp:pitr'" {
    description
      "Augment is valid when LISP role type is ITR or PITR.";
  }
  description
    "This augments LISP devices list with (P)ITR specific
    parameters.";
  container itr {
    presence "LISP (P)ITR operation enabled";
    description
      "ITR parameters";
    container rloc-probing {
      presence "RLOC probing active";
      description
        "RLOC-probing parameters";
      leaf interval {
        type uint16;
        units "seconds";
        description
          "Interval in seconds for resending the probes";
      }
      leaf retries {
        type uint8;
        description
          "Number of retries for sending the probes";
      }
      leaf retries-interval {
        type uint16;
        units "seconds";
        description
          "Interval in seconds between retries when sending probes.
          The action taken if all retries fail to receive is
          impementation specific.";
      }
    }
  }
  leaf itr-rlocs {
    type leafref {
      path "/rt:routing/rt:control-plane-protocols"
        + "/rt:control-plane-protocol/lisp:lisp"
```

```
        + "/lisp:locator-sets/lisp:locator-set"
        + "/lisp:locator-set-name";
    }
    description
        "Reference to a locator set that the (P)ITR includes in
        Map-Requests";
}
container map-resolvers {
    description
        "Map-Resolvers that the (P)ITR uses.";
    leaf-list map-resolver {
        type inet:ip-address;
        description
            "Each Map-Resolver within the list of Map-Resolvers.";
    }
}
container proxy-etr {
    when "../lisp:lisp-role/lisp:lisp-role-type = 'lisp:itr'" {
        description
            "Container exists only when LISP role type is ITR";
    }
    description
        "Proxy ETRs that the ITR uses.";
    leaf-list proxy-etr-address {
        type inet:ip-address;
        description
            "Proxy ETR RLOC address.";
    }
}
container map-cache {
    leaf size {
        type uint32;
        config false;
        description
            "Current number of entries in the EID-to-RLOC map-cache";
    }
    leaf limit {
        type uint32;
        config false;
        description
            "Maximum permissible number of entries in the EID-to-RLOC
            map-cache";
    }
}

uses lisp:mappings;
description
    "EID to RLOCs mappings cache.";
}
```



```

    }
  }
}
<CODE ENDS>

```

4. LISP-ETR Module

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

4.1. Module Structure

```

module: ietf-lisp-etr
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/lisp:lisp:
    +--rw etr!
      +--rw map-servers
        |   +--rw map-server* [ms-address]
        |   |   +--rw ms-address      inet:ip-address
        |   |   +--rw 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
                +--:(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 rlocs?                leafref
+--rw record-ttl?           uint32
+--rw want-map-notify?      boolean
+--rw proxy-reply?          boolean
+--rw registration-interval? uint16

```

4.2. Module Definition

```

<CODE BEGINS> file "ietf-lisp-etr@2018-06-29.yang"
module ietf-lisp-etr {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-etr";

  prefix lisp-etr;

```

```
// RFC Ed.: replace occurrences of XXXX with actual RFC number
// and remove this note
import ietf-lisp {
    prefix lisp;
    reference "RFC XXXX: LISP YANG model";
}
import ietf-lisp-address-types {
    prefix lcaf;
    reference "RFC XXXX: LISP YANG model";
}
import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
}
import ietf-routing {
    prefix "rt";
    reference
        "RFC 8349: A YANG Data Model for Routing Management
        (NMDA version)";
}
```

organization

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

contact

WG Web: <<http://tools.ietf.org/wg/lisp/>>

WG List: <<mailto:lisp@ietf.org>>

Editor: Vina Ermagan
<<mailto:vermagan@cisco.com>>

Editor: Alberto Rodriguez-Natal
<<mailto:natal@cisco.com>>

Editor: Reshad Rahman
<<mailto:rrahman@cisco.com>>";

description

"This YANG module defines the generic parameters for a LISP ETR. The module can be extended by vendors to define vendor-specific parameters and policies.

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

(<http://trustee.ietf.org/license-info>).

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

reference "RFC XXXX";

revision 2018-06-29 {

description

"Initial revision.";

reference

"<https://tools.ietf.org/html/rfc6830>";

}

augment "/rt:routing/rt:control-plane-protocols"

+ "/rt:control-plane-protocol/lisp:lisp" {

when "lisp:lisp-role/lisp:lisp-role-type = 'lisp:etr' or
lisp:lisp-role/lisp:lisp-role-type = 'lisp:petr'" {

description

"Augment is valid when LISP device type is (P)ETR.";

}

description

"This augments LISP devices list with (P)ETR specific
parameters.";

container etr {

presence "LISP (P)ETR operation enabled";

description

"(P)ETR parameters.";

container map-servers {

when "../lisp:lisp-role/lisp:lisp-role-type = 'lisp:etr'" {

description

"Container exists only when LISP device type is ETR.";

}

description

"Map-Servers that the ETR uses.";

list map-server {

key "ms-address";

description

"Each Map-Server within the list of Map-Servers.";

leaf ms-address {

type inet:ip-address;

description

"Map-Server address.";

}

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 lcaf: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 "/rt:routing/rt:control-plane-protocols"
                    + "/rt:control-plane-protocol/lisp:lisp"
            }
        }
    }
}
```

```

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

5. LISP-Map-Server Module

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

5.1. Module Structure

```

module: ietf-lisp-mapserver
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/lisp:lisp:
      +--rw map-server!
        +--rw sites
          |   +--rw site* [site-id]
          |   |   +--rw site-id      uint64
          |   |   +--rw 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-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 site-id*                               uint64
|         +--rw more-specifics-accepted?               boolean
|         +--rw mapping-expiration-timeout?            int16
|         +--ro first-registration-time?
|         |         yang:date-and-time
|         +--ro last-registration-time?
|         |         yang:date-and-time
|         +--rw mapping-records
|         |         +--rw mapping-record* [xtr-id]
|         |         |         +--rw xtr-id
|         |         |         |         lisp:xtr-id-type
|         |         |         +--rw site-id?                uint64
|         |         +--rw eid
|         |         |         +--rw address-type
|         |         |         |         lisp-address-family-ref
|         |         |         +--rw 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 time-to-live?                uint32
+--ro creation-time?
|   yang:date-and-time
+--rw authoritative?              bits
+--rw static?                     boolean
+--rw (locator-list)?
  +--:(negative-mapping)
  |   +--rw map-reply-action?
  |   |   map-reply-action
  +--:(positive-mapping)
  +--rw rlocs
    +--rw locator* [id]
      +--rw id
      |   string
      +--rw locator-address
      |   +--rw address-type
      |   |   lisp-address-family-ref
      +--rw 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?

```

```
| | |  
| | | +--rw priority? uint8  
| | | |   uint8  
| | | +--rw weight?    uint8  
| | | |   uint8  
| | | +--rw multicast-priority?  
| | | |   uint8  
| | | +--rw multicast-weight?  
| | |     uint8  
  
+--ro counters  
  +--ro map-registers-in?           yang:counter64  
  +--ro map-registers-in-auth-failed? yang:counter64  
  +--ro map-notify-records-out?      yang:counter64  
  +--ro proxy-reply-records-out?     yang:counter64  
  +--ro map-requests-forwarded-out? yang:counter64  
+--rw mapping-system-type? lisp:mapping-system-ref  
+--ro summary  
| +--ro number-configured-sites?   uint32  
| +--ro number-registered-sites?   uint32  
| +--ro af-datum  
|   +--ro af-data* [address-type]  
|       +--ro address-type  
|         | lcaf:lisp-address-family-ref  
|       +--ro number-configured-eids?   uint32  
|       +--ro number-registered-eids?   uint32  
+--ro counters  
  +--ro map-registers-in?           yang:counter64  
  +--ro map-registers-in-auth-failed? yang:counter64  
  +--ro map-notify-records-out?      yang:counter64  
  +--ro proxy-reply-records-out?     yang:counter64  
  +--ro map-requests-forwarded-out? yang:counter64
```

5.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp-mapserver@2018-06-29.yang"
module ietf-lisp-mapserver {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver";

  prefix lisp-ms;

  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note
  import ietf-lisp {
    prefix lisp;
    reference "RFC XXXX: LISP YANG model";
  }
}
```



```
import ietf-lisp-address-types {
  prefix lcaf;
  reference "RFC XXXX: LISP YANG model";
}
import ietf-yang-types {
  prefix yang;
  reference "RFC 6991: Common YANG Data Types";
}
import ietf-routing {
  prefix "rt";
  reference
    "RFC 8349: A YANG Data Model for Routing Management
    (NMDA version)";
}

organization
  "IETF LISP (Locator/ID Separation Protocol) Working Group";
contact
  "WG Web:    <http://tools.ietf.org/wg/lisp/>
  WG List:    <mailto:lisp@ietf.org>

  Editor:     Vina Ermagan
               <mailto:vermagan@cisco.com>

  Editor:     Alberto Rodriguez-Natal
               <mailto:natal@cisco.com>

  Editor:     Reshad Rahman
               <mailto:rrahman@cisco.com>";
description
  "This YANG module defines the generic parameters for a LISP
  Map-Server. The module can be extended by vendors to define
  vendor-specific parameters and policies.

  Copyright (c) 2018 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

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

```
reference "RFC XXXX";

revision 2018-06-29 {
  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:counter64;
      description "Number of incoming Map-Register messages";
    }

    leaf map-registers-in-auth-failed {
      type yang:counter64;
      description
        "Number of incoming Map-Register messages failed
        authentication";
    }

    leaf map-notify-records-out {
      type yang:counter64;
      description
        "Number of outgoing Map-Notify records";
    }

    leaf proxy-reply-records-out {
      type yang:counter64;
      description
        "Number of outgoing proxy Map-Reply records";
    }

    leaf map-requests-forwarded-out {
      type yang:counter64;
      description
```

```

        "Number of outgoing Map-Requests forwarded to ETR";
    }
}

augment "/rt:routing/rt:control-plane-protocols"
+ "/rt:control-plane-protocol/lisp:lisp" {
    when "lisp:lisp-role/lisp:lisp-role-type = 'lisp-ms:ms'" {
        description
            "Augment is valid when LISP device type is Map-Server.";
    }
    description
        "This augments LISP devices list with Map-Server specific
        parameters.";
    container map-server {
        presence "LISP Map-Server operation enabled";
        description
            "Map-Server parameters.";
        container sites{
            description
                "Sites to accept registrations from.";
            list site {
                key site-id;
                description
                    "Site that can send registrations.";
                leaf site-id {
                    type uint64;
                    description "Site ID";
                }
                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.";
      }
      leaf first-registration-time {
        type yang:date-and-time;
      }
    }
  }
}
```

```
        config false;
        description
            "Time at which the first registration for this EID
            was received";
    }
    leaf last-registration-time {
        type yang:date-and-time;
        config false;
        description
            "Time at which the last registration for this EID
            was received";
    }
    container mapping-records {
        description
            "Datastore of registered mappings.";
        list mapping-record {
            key xtr-id;
            description
                "Registered mapping.";
            leaf xtr-id {
                type lisp:xtr-id-type;
                description "xTR ID";
            }
            leaf site-id {
                type uint64;
                description "Site ID";
            }
        }
        uses lisp:mapping;
    }
}
}
}
}
}
uses ms-counters;
}
}
leaf mapping-system-type {
    type lisp:mapping-system-ref;
    description
        "A reference to the mapping system";
}

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

    leaf number-configured-sites {
        type uint32;
        description "Number of configured LISP sites";
    }
}
```

```

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

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

```

6. LISP-Map-Resolver Module

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

6.1. Module Structure

```

module: ietf-lisp-mapresolver
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/lisp:lisp:
    +--rw map-resolver!
      +--rw mapping-system-type?    lisp:mapping-system-ref
      +--rw ms-address?             inet:ip-address

```

6.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp-mapresolver@2018-06-29.yang"
module ietf-lisp-mapresolver {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-mapresolver";

  prefix lisp-mr;

  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note
  import ietf-lisp {
    prefix lisp;
    reference "RFC XXXX: LISP YANG model";
  }
  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management
       (NMDA version)";
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/lisp/>
    WG List:  <mailto:lisp@ietf.org>

    Editor:   Vina Ermagan
              <mailto:vermagan@cisco.com>

    Editor:   Alberto Rodriguez-Natal
              <mailto:natal@cisco.com>

    Editor:   Reshad Rahman
              <mailto:rrahman@cisco.com>";
  description
    "This YANG module defines the generic parameters for a LISP
    Map-Resolver. The module can be extended by vendors to define
    vendor-specific parameters and policies.

    Copyright (c) 2018 IETF Trust and the persons identified as
    authors of the code. All rights reserved."
```

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

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

```
reference "RFC XXXX";

revision 2018-06-29 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6833";
}
identity mr {
  base lisp:lisp-role;
  description
    "LISP Map-Resolver.";
}

augment "/rt:routing/rt:control-plane-protocols"
  + "/rt:control-plane-protocol/lisp:lisp" {
  when "lisp:lisp-role/lisp:lisp-role-type = 'lisp-mr:mr'" {
    description
      "Augment is valid when LISP device type is Map-Resolver.";
  }
  description
    "This augments LISP devices list with Map-Resolver specific
    parameters.";
  container map-resolver {
    presence "LISP Map-Resolver operation enabled";
    description
      "Map-Resolver parameters.";
    leaf mapping-system-type {
      type lisp:mapping-system-ref;
      description
        "A reference to the mapping system";
    }
    leaf ms-address {
      when "../mapping-system-type='lisp:single-node-mapping-system'";
      type inet:ip-address;
      description
        "address to reach the Map Server when "
```



```
        + "lisp-mr:single-node-mapping-system is being used.";
    }
}
}
<CODE ENDS>
```

7. LISP-Address-Types Module

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

7.1. Module Definition

```
<CODE BEGINS> file "ietf-lisp-address-types@2018-06-29.yang"
module ietf-lisp-address-types {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types";

  prefix laddr;

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types";
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "WG Web:    <http://tools.ietf.org/wg/lisp/>
    WG List:    <mailto:lisp@ietf.org>

    Editor:     Vina Ermagan
                <mailto:vermagan@cisco.com>

    Editor:     Alberto Rodriguez-Natal
                <mailto:natal@cisco.com>

    Editor:     Reshad Rahman
                <mailto:rrahman@cisco.com>";
  description
    "This YANG module defines the LISP Canonical Address Formats
    (LCAF) for LISP. The module can be extended by vendors to
```

define vendor-specific parameters.

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```

";
// RFC Ed.: replace XXXX with actual RFC number and remove
// this note
reference "RFC XXXX";

revision 2018-06-29 {
  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 lcac {
    base lisp-address-family;
    description
      "IANA LISP Canonical Address Format address family.";
  }
  identity null-address-lcac {
    base lcac;
    description
      "Null body LCAF type.";
  }
  identity afi-list-lcac {
    base lcac;
    description
      "AFI-List LCAF type.";
  }
  identity instance-id-lcac {
    base lcac;
    description
      "Instance-ID LCAF type.";
  }
  identity as-number-lcac {
    base lcac;
    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-address-afi'" {
                description
                    "When AFI is 0.";
            }
            type empty;
            description

```

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

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



```
        description
          "When LCAF type is Instance-ID";
      }
      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>
```

7.2. Data Model examples

This section presents some simple and illustrative examples on how to configure LISP.

7.2.1. LISP protocol instance

The following is an example configuration for a LISP protocol instance with the name "LISP1". There are also 2 VNIs configured.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <network-instances xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
    <network-instance>
      <name>VRF-BLUE</name>
      <vrf-root/>
      <enabled>true</enabled>
    </network-instance>
    <network-instance>
      <name>VRF-RED</name>
      <vrf-root/>
      <enabled>true</enabled>
    </network-instance>
  </network-instances>
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
          lisp:lisp
        </type>
        <name>LISP1</name>
        <lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
          <virtual-networks>
            <virtual-network>
              <vni>1000</vni>
              <ni-name>VRF-BLUE</ni-name>
            </virtual-network>
            <virtual-network>
              <vni>2000</vni>
              <ni-name>VRF-RED</ni-name>
            </virtual-network>
          </virtual-networks>
        </lisp>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>
```


7.2.2. LISP ITR

The following is an example configuration for ITR functionality under "LISP1". There are 2 Map-Resolvers configured.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
          lisp:lisp
        </type>
        <name>LISP1</name>
        <lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
          <lisp-role>
            <lisp-role-type>itr</lisp-role-type>
          </lisp-role>
          <itr xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp-itr">
            <map-resolvers>
              <map-resolver>2001:db8:203:0:113::1</map-resolver>
              <map-resolver>2001:db8:204:0:113::1</map-resolver>
            </map-resolvers>
          </itr>
        </lisp>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>
```

7.2.3. LISP ETR

The following is an example configuration for ETR functionality under "LISP1". There are 2 Map-Servers and 2 local EIDs configured.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
          lisp:lisp
        </type>
        <name>LISP1</name>
```

```

<lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
  <lisp-role>
    <lisp-role-type>etr</lisp-role-type>
  </lisp-role>
  <lisp-router-id>
    <site-id>1</site-id>
  </lisp-router-id>
  <etr xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp-etr">
    <map-servers>
      <map-server>
        <ms-address>2001:db8:203:0:113::1</ms-address>
        <auth-key>*Kye^$$1#gb91U04zpa!</auth-key>
        <auth-key-type>hmac-sha-256-128</auth-key-type>
      </map-server>
      <map-server>
        <ms-address>2001:db8:204:0:113::1</ms-address>
        <auth-key>*Kye^$$1#gb91U04zpa!</auth-key>
        <auth-key-type>hmac-sha-256-128</auth-key-type>
      </map-server>
    </map-servers>
    <local-eids>
      <virtual-network>
        <vni>1000</vni>
        <eids>
          <local-eid>
            <id>2001:db8:400:0:100::0</id>
            <eid-address>
              <address-type xmlns:laddr=
                "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
                laddr:ipv6-prefix-afi
              </address-type>
              <ipv6-prefix>2001:db8:400:0:100::/80</ipv6-prefix>
            </eid-address>
          </local-eid>
        </eids>
      </virtual-network>
      <virtual-network>
        <vni>2000</vni>
        <eids>
          <local-eid>
            <id>2001:db8:800:0:200::0</id>
            <eid-address>
              <address-type xmlns:laddr=
                "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
                laddr:ipv6-prefix-afi
              </address-type>
              <ipv6-prefix>2001:db8:800:0:200::/80</ipv6-prefix>
            </eid-address>
          </local-eid>
        </eids>
      </virtual-network>
    </local-eids>
  </etr>
</lisp>

```

```

        </local-eid>
      </eids>
    </virtual-network>
  </local-eids>
</etr>
</lisp>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</config>

```

7.2.4. LISP Map-Server

The following is an example configuration for Map-Server functionality under "LISP1". There are 2 mappings configured.

```

<config xmlns="http://tail-f.com/ns/config/1.0">
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
          lisp:lisp
        </type>
        <name>LISP1</name>
        <lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
          <lisp-role>
            <lisp-role-type xmlns:lisp-ms=
              "urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver">
              lisp-ms:ms
            </lisp-role-type>
          </lisp-role>
          <map-server xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver">
            <sites>
              <site>
                <site-id>1</site-id>
                <auth-key>
                  <auth-key-value>*Kye^$$1#gb91U04zpa!</auth-key-value>
                  <auth-key-type>hmac-sha-256-128</auth-key-type>
                </auth-key>
              </site>
            </sites>
            <virtual-network-ids>
              <virtual-network-identifier>
                <vni>1000</vni>
                <mappings>

```

```
<mapping>
  <eid-id>1</eid-id>
  <eid-address>
    <address-type xmlns:laddr=
      "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
      laddr:ipv6-prefix-afi
    </address-type>
    <ipv6-prefix>2001:db8:400:0:100::/80</ipv6-prefix>
  </eid-address>
</mapping>
</mappings>
</virtual-network-identifier>
<virtual-network-identifier>
  <vni>2000</vni>
  <mappings>
    <mapping>
      <eid-id>1</eid-id>
      <eid-address>
        <address-type xmlns:laddr=
          "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
          laddr:ipv6-prefix-afi
        </address-type>
        <ipv6-prefix>2001:db8:800:0:200::/80</ipv6-prefix>
      </eid-address>
    </mapping>
  </mappings>
</virtual-network-identifier>
</virtual-network-ids>
</map-server>
</lisp>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</config>
```

8. Acknowledgments

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

9. IANA Considerations

The IANA is requested to assign a new namespace URI from the IETF XML registry.

This document registers the following namespace URIs in the IETF XML registry [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-lisp

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-itr

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-etr

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-mapresolver

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-address-types

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

10. Security Considerations

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, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <<https://www.rfc-editor.org/info/rfc6832>>.
- [RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, DOI 10.17487/RFC6833, January 2013, <<https://www.rfc-editor.org/info/rfc6833>>.

- [RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, DOI 10.17487/RFC6836, January 2013, <<https://www.rfc-editor.org/info/rfc6836>>.
- [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.
- [RFC8111] Fuller, V., Lewis, D., Ermagan, V., Jain, A., and A. Smirnov, "Locator/ID Separation Protocol Delegated Database Tree (LISP-DDT)", RFC 8111, DOI 10.17487/RFC8111, May 2017, <<https://www.rfc-editor.org/info/rfc8111>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.

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

LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: 28 August 2022

V. Ermagan
Google
A. Rodriguez-Natal
F. Coras
Cisco Systems
C. Moberg
Avassa
R. Rahman

A. Cabellos-Aparicio
Technical University of Catalonia
F. Maino
Cisco Systems
24 February 2022

LISP YANG Model
draft-ietf-lisp-yang-17

Abstract

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

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA).

Status of This Memo

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

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

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

This Internet-Draft will expire on 28 August 2022.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
1.2. Tree Diagrams	3
1.3. Prefixes	3
2. LISP Module	4
2.1. Module Structure	4
2.2. Module Definition	7
3. LISP-ITR Module	19
3.1. Module Structure	19
3.2. Module Definition	24
4. LISP-ETR Module	28
4.1. Module Structure	28
4.2. Module Definition	30
5. LISP-Map-Server Module	34
5.1. Module Structure	34
5.2. Module Definition	43
6. LISP-Map-Resolver Module	49
6.1. Module Structure	50
6.2. Module Definition	50
7. LISP-Address-Types Module	52
7.1. Module Definition	52
7.2. Data Model examples	67
7.2.1. LISP protocol instance	67
7.2.2. LISP ITR	69
7.2.3. LISP ETR	69
7.2.4. LISP Map-Server	72
8. Acknowledgments	73
9. IANA Considerations	73
10. Security Considerations	76
11. Normative References	79
Authors' Addresses	82

1. Introduction

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

1.1. Requirements Language

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

1.2. Tree Diagrams

This document uses the graphical representation of data models defined in [RFC8340].

1.3. Prefixes

The table below provides a summary of the prefixes used by this document.

Prefix	YANG module	Reference
lisp	ietf-lisp	Section 2
if	ietf-interfaces	[RFC8343]
lisp-at	ietf-lisp-address-types	Section 7
yang	ietf-yang-types	[RFC6991]
rt	ietf-routing	[RFC8022]
ni	ietf-network-instance	[RFC8529]
lisp-itr	ietf-lisp-itr	Section 3
inet	ietf-inet-types	[RFC6991]
lisp-etr	ietf-lisp-etr	Section 4
lisp-ms	ietf-lisp-mapserver	Section 5
lisp-mr	ietf-lisp-mapresolver	Section 6

Table 1: Prefixes and corresponding YANG modules

2. LISP Module

This is the base LISP module. It is further augmented by the LISP device role specific modules defined elsewhere in this document.

2.1. Module Structure

```

module: ietf-lisp
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol:
      +--rw lisp
        +--rw locator-sets
          +--rw locator-set* [locator-set-name]
            +--rw locator-set-name      string
            +--rw (locator-type)?
              +--:(local-interface)
                +--rw interface* [interface-ref]
                  +--rw interface-ref    if:interface-ref
                  +--rw priority?         uint8
                  +--rw weight?          uint8

```

```

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

```

```

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

```

```

|                                     |
|                                     | +--rw source?      simple-address
|                                     | +--rw dest?       simple-address
|                                     | +---:(key-value-address)
|                                     |   +--rw key-value-address
|                                     |   +--rw key?        simple-address
|                                     |   +--rw value?      simple-address
|                                     | +---:(service-path)
|                                     |   +--rw service-path
|                                     |   +--rw service-path-id?
|                                     |     |
|                                     |     | service-path-id-type
|                                     |     +--rw service-index?    uint8
|                                     |
+---rw priority?                uint8
+---rw weight?                  uint8
+---rw multicast-priority?      uint8
+---rw multicast-weight?        uint8
+---rw lisp-role* [lisp-role-type]
|   +--rw lisp-role-type      lisp-role-ref
+---rw lisp-router-id
|   +--rw site-id?           uint64
|   +--rw xtr-id?            lisp:xtr-id-type
+---rw vpns
|   +--rw vpn* [instance-id]
|     +--rw instance-id      lisp-at:instance-id-type
|     +--rw iid-name
|       -> /ni:network-instances/network-instance/name

```

2.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp@2021-02-22.yang"
module ietf-lisp {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp";

  prefix lisp;

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }
  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note
  import ietf-lisp-address-types {
    prefix lisp-at;
    reference "RFC XXXX: LISP YANG model";
  }
}
```

```
}
import ietf-yang-types {
  prefix yang;
  reference "RFC 6991: Common YANG Data Types";
}
import ietf-routing {
  prefix "rt";
  reference
    "RFC 8349: A YANG Data Model for Routing Management
    (NMDA version)";
}
import ietf-network-instance {
  prefix "ni";
  reference
    "RFC 8529: YANG Model for Network Instances";
}

organization
  "IETF LISP (Locator/ID Separation Protocol) Working Group";
contact
  "WG Web:  <http://tools.ietf.org/wg/lisp/>
  WG List:  <mailto:lisp@ietf.org>

  Editor:    Vina Ermagan
             <mailto:ermagan@gmail.com>

  Editor:    Alberto Rodriguez-Natal
             <mailto:natal@cisco.com>

  Editor:    Reshad Rahman
             <mailto:reshad@yahoo.com>";

description
  "This YANG module defines the generic parameters for LISP.
  The module can be extended by vendors to define vendor-specific
  LISP parameters and policies.

  Copyright (c) 2018 IETF Trust and the persons identified as
  authors of the code.  All rights reserved.

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

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



```
    ";

    reference "RFC XXXX";

    revision 2021-02-22 {
      description
        "Initial revision.";
      reference
        "https://tools.ietf.org/html/rfc6830";
    }

    /*
     * Identity definitions
     */
    identity lisp {
      base "rt:control-plane-protocol";
      description "LISP protocol.";
      reference
        "RFC 6830: The Locator/ID Separation Protocol (LISP).";
    }

    identity lisp-role {
      description
        "LISP router role.";
    }
    identity itr {
      base lisp-role;
      description
        "LISP ITR.";
    }
    identity pitr {
      base lisp-role;
      description
        "LISP PITR.";
    }
    identity etr {
      base lisp-role;
      description
        "LISP ETR.";
    }
    identity petr {
      base lisp-role;
      description
        "LISP PETR.";
    }

    identity mapping-system {
      description
```

```
    "Mapping System interface";
  }
  identity single-node-mapping-system {
    base mapping-system;
    description
      "logically singular Map Server";
  }

  identity map-reply-act {
    description
      "Defines the lisp map-cache ACT type";
    reference
      "https://www.iana.org/assignments/lisp-parameters"
      + "/lisp-parameters.xhtml#lisp-act-value";
  }
  identity no-action {
    base map-reply-act;
    description
      "Mapping is kept alive and no encapsulation
      occurs.";
  }
  identity natively-forward {
    base map-reply-act;
    description
      "Matching packets are not encapsulated or
      dropped but natively forwarded.";
  }
  identity send-map-request {
    base map-reply-act;
    description
      "Matching packets invoke Map-Requests.";
  }
  identity drop-no-reason {
    base map-reply-act;
    description
      "Matching packets are dropped.";
  }
  identity drop-policy-denied {
    base map-reply-act;
    description
      "Matching packets are dropped (due to policy).";
  }
  identity drop-auth-failure {
    base map-reply-act;
    description
      "Matching packets are dropped (due to authentication
      failure).";
  }
}
```

```
identity auth-algorithm {
  description
    "Base identity for the authentication mechanisms supported by
    LISP.";
  reference
    "https://www.iana.org/assignments/lisp-parameters"
    + "/lisp-parameters.xhtml#lisp-key-id-numbers";
}
identity no-auth-algorithm {
  base auth-algorithm;
  description
    "No authentication.";
}
identity hmac-sha-1-96-none {
  base auth-algorithm;
  description
    "MAC = HMAC-SHA-1-96 (RFC2404), KDF = none";
}
identity hmac-sha-256-128-none {
  base auth-algorithm;
  description
    "MAC = HMAC-SHA-256-128 (RFC4868), KDF = none";
}
identity hmac-sha-256-128-HKDF-SHA2562 {
  base auth-algorithm;
  description
    "MAC = HMAC-SHA-256-128, KDF = HKDF-SHA2562 (RFC4868)";
}

typedef mapping-system-ref {
  type identityref {
    base mapping-system;
  }
  description
    "Mapping System reference";
}

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

typedef map-reply-action {
  type identityref {
    base map-reply-act;
  }
}
```

```
    description
      "Map-Reply action reference";
  }
  typedef eid-id {
    type string {
      pattern '[a-zA-Z0-9\-\_\.]*';
    }
    description
      "Type encoding of lisp-addresses to be generally used in EID
       keyed lists.";
  }
  typedef auth-algorithm-type {
    type identityref {
      base auth-algorithm;
    }
    description
      "Authentication algorithm reference";
  }
  typedef xtr-id-type {
    type binary {
      length "16";
    }
    description
      "128-bit xTR identifier.";
  }

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

```
    }
  }

  grouping locators-grouping {
    description
      "Grouping that defines a list of LISP locators.";
    list locator {
      key "locator-id";
      description
        "List of routing locators";
      leaf locator-id {
        type string {
          length "1..64";
          pattern '[a-zA-Z0-9\\-\\.:]*';
        }
        description
          "Locator id";
      }
      container locator-address {
        uses lisp-at:lisp-address;
        description
          "The locator address provided in LISP canonincaal
          address format.";
      }
      uses locator-properties;
    }
  }

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

  grouping mapping {
    description
      "Grouping that defines a LISP mapping.";
```

```
container eid {
  uses lisp-at:lisp-address;
  description
    "End-host Identifier (EID) to be mapped to a list of
    locators";
}
leaf time-to-live {
  type uint32;
  units minutes;
  description
    "Mapping validity period in minutes (as per RF6830).";
}
leaf creation-time {
  type yang:date-and-time;
  config false;
  description
    "Time when the mapping was created.";
}
leaf authoritative {
  type bits {
    bit A {
      description
        "Authoritative bit.";
    }
  }
  description
    "Bit that indicates if mapping comes from an
    authoritative source.";
}
leaf static {
  type boolean;
  default "false";
  description
    "This leaf should be true if the mapping is static.";
}
choice locator-list {
  description
    "list of locartors are either negative, or positive.";
  case negative-mapping {
    leaf map-reply-action {
      type map-reply-action;
      description
        "Forwarding action for a negative mapping.";
    }
  }
  case positive-mapping {
    container rlocs {
      uses locators-grouping;
    }
  }
}
```

```
        description
            "List of locators for a positive mapping.";
    }
}
}

grouping mappings {
    description
        "Grouping that defines a list of LISP mappings.";
    list vpn {
        key "instance-id";
        description
            "VPN to which the mappings belong.";
        leaf instance-id {
            type leafref {
                path "/rt:routing/rt:control-plane-protocols"
                    + "/rt:control-plane-protocol/lisp:lisp"
                    + "/lisp:vpns/lisp:vpn"
                    + "/lisp:instance-id";
            }
            description
                "VPN identifier.";
        }
        container mappings {
            description
                "Mappings within the VPN.";
            list mapping {
                key "eid-id";
                description
                    "List of EID to RLOCs mappings.";
                leaf eid-id {
                    type eid-id;
                    description
                        "Id that uniquely identifies a mapping.";
                }
                uses mapping;
            }
        }
    }
}

grouping auth-key {
    description "Grouping that defines authentication keys.";
    container authentication-keys {
        description "Multiple authentication keys can be defined.";
        list authentication-key {
            key "auth-key-id";
```

```
    description
    "Authentication key parameters.";
    leaf auth-key-id {
      type string {
        pattern '[a-zA-Z0-9\-\_\.]*';
      }
      description
      "Identifier of the authentication key.";
    }
    leaf-list auth-algorithm-id {
      type lisp:auth-algorithm-type;
      description
      "Authentication algorithm used with the key.";
    }
    leaf auth-key-value {
      type string;
      description
      "Clear text authentication key.";
    }
  }
}

augment "/rt:routing/rt:control-plane-protocols"
+ "/rt:control-plane-protocol" {
  when "derived-from-or-self(rt:type, 'lisp:lisp')" {
    description
    "This augmentation is only valid for a control-plane protocol
    instance of LISP.";
  }
  description "LISP protocol ietf-routing module
  control-plane-protocol augmentation.";

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

    container locator-sets {
      description
      "Container that defines a named locator set which can be
      referenced elsewhere.";
      list locator-set {
        key "locator-set-name";
        description
        "Multiple locator sets can be defined.";
        leaf locator-set-name {
          type string {
            length "1..64";
          }
        }
      }
    }
  }
}
```



```
        pattern '[a-zA-Z0-9\-\_\.]*';
    }
    description
        "Locator set name";
}
choice locator-type {
    description
        "Locator sets can be based on local interfaces, or
        general locators.";
    case local-interface {
        uses local-locators-grouping;
        description
            "List of locators in this set based on local
            interfaces.";
    }
    case general-locator {
        uses locators-grouping;
        description
            "List of locators in this set based on
            lisp-address.";
    }
}
}
}

list lisp-role {
    key lisp-role-type;
    description
        "List of lisp device roles such as MS, MR, ITR,
        PITR, ETR or PETR.";
    leaf lisp-role-type {
        type lisp-role-ref;
        description
            "The type of LISP device - identity derived from the
            'lisp-device' base identity.";
    }
}

container lisp-router-id {
    when "../lisp-role/lisp-role-type = 'lisp:itr' or
        ../lisp-role/lisp-role-type = 'lisp:pitr' or
        ../lisp-role/lisp-role-type = 'lisp:etr' or
        ../lisp-role/lisp-role-type = 'lisp:petr'" {
        description "Only when ITR, PITR, ETR or PETR.";
    }
    description
        "Site-ID and xTR-ID of the device.";
    leaf site-id {
```

```

    type uint64;
    description "Site ID";
}
leaf xtr-id {
    type lisp:xtr-id-type;
    description "xTR ID";
}
}

container vpns {
    when "../lisp-role/lisp-role-type = 'lisp:itr' or
        ../lisp-role/lisp-role-type = 'lisp:pitr' or
        ../lisp-role/lisp-role-type = 'lisp:etr' or
        ../lisp-role/lisp-role-type = 'lisp:petr'" {
        description "Only when ITR, PITR, ETR or PETR.";
    }
    description "VPNs";
    list vpn {
        key instance-id;
        unique "iid-name";
        description "List of VPNs";

        leaf instance-id {
            type lisp-at:instance-id-type;
            description
                "VPN identifier. The value 0 for instance-id must be
                 used for the default VRF.";
        }
        leaf iid-name {
            type leafref {
                path "/ni:network-instances/ni:network-instance"
                    + "/ni:name";
            }
            mandatory true;
            description
                "Name of VPN (e.g. VRF) to which an instance-id is
                 bound. Each instance-id is bound to a different VPN";
        }
    }
}
}
}

<CODE ENDS>
```

3. LISP-ITR Module

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

3.1. Module Structure

```

module: ietf-lisp-itr
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/lisp:lisp:
      +--rw itr!
        +--rw rloc-probing!
          |   +--rw interval?          uint16
          |   +--rw retries?           uint8
          |   +--rw retries-interval?  uint16
        +--rw itr-rlocs?      leafref
        +--rw map-resolvers
          |   +--rw map-resolver*      inet:ip-address
        +--rw proxy-etrs
          |   +--rw proxy-etr-address*  inet:ip-address
        +--rw map-cache
          +--ro size?      uint32
          +--ro limit?     uint32
          +--rw vpn* [instance-id]
            +--rw instance-id      leafref
            +--rw mappings
              +--rw mapping* [eid-id]
                +--rw eid-id          eid-id
                +--rw eid
                  +--rw address-type
                  |   +--rw lisp-address-family-ref
                  +--rw (address)?
                    +--:(no-address)
                    |   +--rw no-address?          empty
                    +--:(ipv4)
                    |   +--rw ipv4?
                    |       inet:ipv4-address
                    +--:(ipv4-prefix)
                    |   +--rw ipv4-prefix?
                    |       inet:ipv4-prefix
                    +--:(ipv6)
                    |   +--rw ipv6?
                    |       inet:ipv6-address
                    +--:(ipv6-prefix)
                    |   +--rw ipv6-prefix?
                    |       inet:ipv6-prefix
                    +--:(mac)
                    |   +--rw mac?

```

```

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

```

```

|                                     simple-address
+---:(nat-traversal)
|   +---rw nat-traversal
|   |   +---rw ms-udp-port?          uint16
|   |   +---rw etr-udp-port?        uint16
|   |   +---rw global-etr-rloc?
|   |   |       simple-address
|   |   +---rw ms-rloc?
|   |   |       simple-address
|   |   +---rw private-etr-rloc?
|   |   |       simple-address
|   |   +---rw rtr-rlocs*
|   |       simple-address
+---:(explicit-locator-path)
|   +---rw explicit-locator-path
|   |   +---rw hop* [hop-id]
|   |   |       +---rw hop-id        string
|   |   |       +---rw address?      simple-address
|   |   |       +---rw lrs-bits?     bits
+---:(source-dest-key)
|   +---rw source-dest-key
|   |   +---rw source?      simple-address
|   |   +---rw dest?       simple-address
+---:(key-value-address)
|   +---rw key-value-address
|   |   +---rw key?        simple-address
|   |   +---rw value?     simple-address
+---:(service-path)
|   +---rw service-path
|   |   +---rw service-path-id?
|   |   |       service-path-id-type
|   |   +---rw service-index?    uint8
+---rw time-to-live?          uint32
+---ro creation-time?         yang:date-and-time
+---rw authoritative?        bits
+---rw static?                boolean
+---rw (locator-list)?
|   +---:(negative-mapping)
|   |   +---rw map-reply-action?    map-reply-action
+---:(positive-mapping)
|   +---rw rlocs
|   |   +---rw locator* [locator-id]
|   |   |       +---rw locator-id    string
|   |   |       +---rw locator-address
|   |   |       |       +---rw address-type
|   |   |       |       |       lisp-address-family-ref
|   |   |       +---rw (address)?
|   |   |       |       +---:(no-address)

```

```

    +---rw no-address?
        empty
+---:(ipv4)
    +---rw ipv4?
        inet:ipv4-address
+---:(ipv4-prefix)
    +---rw ipv4-prefix?
        inet:ipv4-prefix
+---:(ipv6)
    +---rw ipv6?
        inet:ipv6-address
+---:(ipv6-prefix)
    +---rw ipv6-prefix?
        inet:ipv6-prefix
+---:(mac)
    +---rw mac?
        yang:mac-address
+---:(distinguished-name)
    +---rw distinguished-name?
        distinguished-name-type
+---:(as-number)
    +---rw as-number?
        inet:as-number
+---:(null-address)
    +---rw null-address
    +---rw address?    empty
+---:(afi-list)
    +---rw afi-list
    +---rw address-list*
        simple-address
+---:(instance-id)
    +---rw instance-id
    +---rw instance-id?
        | instance-id-type
    +---rw mask-length?    uint8
    +---rw address?
        simple-address
+---:(as-number-lcaf)
    +---rw as-number-lcaf
    +---rw as?
        | inet:as-number
    +---rw address?
        simple-address
+---:(application-data)
    +---rw application-data
    +---rw address?
        | simple-address
    +---rw protocol?

```

```

|         uint8
+--rw ip-tos?
|         int32
+--rw local-port-low?
|         inet:port-number
+--rw local-port-high?
|         inet:port-number
+--rw remote-port-low?
|         inet:port-number
+--rw remote-port-high?
|         inet:port-number
+--:(geo-coordinates)
+--rw geo-coordinates
+--rw latitude?
|         bits
+--rw latitude-degrees?
|         uint8
+--rw latitude-minutes?
|         uint8
+--rw latitude-seconds?
|         uint8
+--rw longitude?
|         bits
+--rw longitude-degrees?
|         uint16
+--rw longitude-minutes?
|         uint8
+--rw longitude-seconds?
|         uint8
+--rw altitude?
|         int32
+--rw address?
|         simple-address
+--:(nat-traversal)
+--rw nat-traversal
+--rw ms-udp-port?
|         uint16
+--rw etr-udp-port?
|         uint16
+--rw global-etr-rloc?
|         simple-address
+--rw ms-rloc?
|         simple-address
+--rw private-etr-rloc?
|         simple-address
+--rw rtr-rlocs*
|         simple-address
+--:(explicit-locator-path)

```

```

+--rw explicit-locator-path
+--rw hop* [hop-id]
+--rw hop-id
|   string
+--rw address?
|   simple-address
+--rw lrs-bits?   bits
+--:(source-dest-key)
+--rw source-dest-key
+--rw source?
|   simple-address
+--rw dest?
|   simple-address
+--:(key-value-address)
+--rw key-value-address
+--rw key?
|   simple-address
+--rw value?
|   simple-address
+--:(service-path)
+--rw service-path
+--rw service-path-id?
|   service-path-id-type
+--rw service-index?
|   uint8
+--rw priority?           uint8
+--rw weight?             uint8
+--rw multicast-priority? uint8
+--rw multicast-weight?   uint8

```

3.2. Module Definition

```

<CODE BEGINS> file "ietf-lisp-itr@2019-02-23.yang"
module ietf-lisp-itr {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-itr";

  prefix lisp-itr;

  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note
  import ietf-lisp {
    prefix lisp;
    reference "RFC XXXX: LISP YANG model";
  }
  import ietf-inet-types {
    prefix inet;
  }

```



```
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management
      (NMDA version)";
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/lisp/>
    WG List:  <mailto:lisp@ietf.org>

    Editor:   Vina Ermagan
              <mailto:ermagan@gmail.com>

    Editor:   Alberto Rodriguez-Natal
              <mailto:natal@cisco.com>

    Editor:   Reshad Rahman
              <mailto:reshad@yahoo.com>";
  description
    "This YANG module defines the generic parameters for a LISP
    ITR. The module can be extended by vendors to define
    vendor-specific parameters and policies.

    Copyright (c) 2018 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

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

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

  reference "RFC XXXX";

  revision 2019-02-23 {
    description
      "Initial revision.";
    reference
```

```
    "https://tools.ietf.org/html/rfc6830";
  }
  augment "/rt:routing/rt:control-plane-protocols"
    + "/rt:control-plane-protocol/lisp:lisp" {
    when "lisp:lisp-role/lisp:lisp-role-type = 'lisp:itr' or
        lisp:lisp-role/lisp:lisp-role-type = 'lisp:pitr'" {
      description
        "Augment is valid when LISP role type is ITR or PITR.";
    }
    description
      "This augments the LISP devices list with (P)ITR specific
      parameters.";
    container itr {
      presence "LISP (P)ITR operation enabled";
      description
        "ITR parameters";
      container rloc-probing {
        presence "RLOC probing active";
        description
          "RLOC-probing parameters";
        leaf interval {
          type uint16;
          units "seconds";
          description
            "Interval in seconds for resending the probes";
        }
        leaf retries {
          type uint8;
          description
            "Number of retries for sending the probes";
        }
        leaf retries-interval {
          type uint16;
          units "seconds";
          description
            "Interval in seconds between retries when sending probes.
            The action taken if all retries fail to receive is
            implementation specific.";
        }
      }
    }
    leaf itr-rlocs {
      type leafref {
        path "/rt:routing/rt:control-plane-protocols"
          + "/rt:control-plane-protocol/lisp:lisp"
          + "/lisp:locator-sets/lisp:locator-set"
          + "/lisp:locator-set-name";
      }
      description

```

```
        "Reference to a locator set that the (P)ITR includes in
        Map-Requests";
    }
    container map-resolvers {
        description
            "Map-Resolvers that the (P)ITR uses.";
        leaf-list map-resolver {
            type inet:ip-address;
            description
                "Each Map-Resolver within the list of Map-Resolvers.";
        }
    }
    container proxy-etr {
        when "../lisp:lisp-role/lisp:lisp-role-type='lisp:itr'" {
            description
                "Container exists only when LISP role type is ITR";
        }
        description
            "Proxy ETRs that the ITR uses.";
        leaf-list proxy-etr-address {
            type inet:ip-address;
            description
                "Proxy ETR RLOC address.";
        }
    }
    container map-cache {
        leaf size {
            type uint32;
            config false;
            description
                "Current number of entries in the EID-to-RLOC map-cache";
        }
        leaf limit {
            type uint32;
            config false;
            description
                "Maximum permissible number of entries in the EID-to-RLOC
                map-cache";
        }
    }

    uses lisp:mappings;
    description
        "EID to RLOCs mappings cache.";
}
}
}
}
<CODE ENDS>
```

4. LISP-ETR Module

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

4.1. Module Structure

```

module: ietf-lisp-etr
augment /rt:routing/rt:control-plane-protocols
  /rt:control-plane-protocol/lisp:lisp:
  +--rw etr!
    +--rw map-servers
      +--rw map-server* [ms-address]
        +--rw ms-address          inet:ip-address
        +--rw authentication-keys
          +--rw authentication-key* [auth-key-id]
            +--rw auth-key-id      string
            +--rw auth-algorithm-id*
              | lisp:auth-algorithm-type
            +--rw auth-key-value?  string
    +--rw local-eids
      +--rw vpn* [instance-id]
        +--rw instance-id      leafref
        +--rw eids
          +--rw local-eid* [eid-id]
            +--rw eid-id          lisp:eid-id
            +--rw eid-address
              +--rw address-type
              | lisp-address-family-ref
            +--rw (address)?
              +--:(no-address)
              | +--rw no-address?          empty
              +--:(ipv4)
              | +--rw ipv4?
              |   inet:ipv4-address
              +--:(ipv4-prefix)
              | +--rw ipv4-prefix?
              |   inet:ipv4-prefix
              +--:(ipv6)
              | +--rw ipv6?
              |   inet:ipv6-address
              +--:(ipv6-prefix)
              | +--rw ipv6-prefix?
              |   inet:ipv6-prefix
              +--:(mac)
              | +--rw mac?
              |   yang:mac-address
            +--:(distinguished-name)

```

```

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

```

```

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

```

4.2. Module Definition

```

<CODE BEGINS> file "ietf-lisp-etr@2021-02-22.yang"
module ietf-lisp-etr {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-etr";

  prefix lisp-etr;

  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note

```

```
import ietf-lisp {
  prefix lisp;
  reference "RFC XXXX: LISP YANG model";
}
import ietf-lisp-address-types {
  prefix lisp-at;
  reference "RFC XXXX: LISP YANG model";
}
import ietf-inet-types {
  prefix inet;
  reference "RFC 6991: Common YANG Data Types";
}
import ietf-routing {
  prefix "rt";
  reference
    "RFC 8349: A YANG Data Model for Routing Management
    (NMDA version)";
}

organization
  "IETF LISP (Locator/ID Separation Protocol) Working Group";
contact
  "WG Web:  <http://tools.ietf.org/wg/lisp/>
  WG List:  <mailto:lisp@ietf.org>

  Editor:    Vina Ermagan
             <mailto:ermagan@gmail.com>

  Editor:    Alberto Rodriguez-Natal
             <mailto:natal@cisco.com>

  Editor:    Reshad Rahman
             <mailto:reshad@yahoo.com>;

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

  Copyright (c) 2018 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

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

reference "RFC XXXX";

revision 2021-02-22 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6830";
}
augment "/rt:routing/rt:control-plane-protocols"
  + "/rt:control-plane-protocol/lisp:lisp" {
  when "lisp:lisp-role/lisp:lisp-role-type = 'lisp:etr' or
    lisp:lisp-role/lisp:lisp-role-type = 'lisp:petr'" {
    description
      "Augment is valid when LISP device type is (P)ETR.";
  }
  description
    "This augments the LISP devices list with (P)ETR specific
    parameters.";
  container etr {
    presence "LISP (P)ETR operation enabled";
    description
      "(P)ETR parameters.";

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

  container local-eids {
```



```
when "../../../lisp:lisp-role/lisp:lisp-role-type='lisp:etr'" {
  description
    "Container exists only when LISP device type is ETR.";
}
description
  "VPNs served by the ETR.";
list vpn {
  key "instance-id";
  description
    "VPN for local-EIDs.";
  leaf instance-id {
    type leafref {
      path "/rt:routing/rt:control-plane-protocols"
        + "/rt:control-plane-protocol/lisp:lisp"
        + "/lisp:vpns/lisp:vpn"
        + "/lisp:instance-id";
    }
    description
      "VPN identifier.";
  }
  container eids {
    description
      "EIDs served by the ETR.";
    list local-eid {
      key "eid-id";
      description
        "List of local EIDs.";
      leaf eid-id {
        type lisp:eid-id;
        description
          "Unique id of local EID.";
      }
      container eid-address {
        uses lisp-at:lisp-address;
        description
          "EID address in generic LISP address format.";
      }
      leaf rlocs {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols"
            + "/rt:control-plane-protocol/lisp:lisp"
            + "/lisp:locator-sets/lisp:locator-set"
            + "/lisp:locator-set-name";
        }
        description
          "Locator set mapped to this local EID.";
      }
      leaf record-ttl {
```

```

        type uint32;
        units minutes;
        description
            "Validity period of the EID to RLOCs mapping
             provided in Map-Replies.";
    }
    leaf want-map-notify {
        type boolean;
        default "true";
        description
            "Flag which if set in a Map-Register requests that
             a Map-Notify be sent in response.";
    }
    leaf proxy-reply {
        type boolean;
        default "false";
        description
            "Flag which if set in a Map-Register requests that
             the Map-Server proxy Map-Replies for the ETR.";
    }
    leaf registration-interval {
        type uint16;
        units "seconds";
        default "60";
        description
            "Interval between consecutive Map-Registers.";
    }
}
}
}
}
}
}
}
<CODE ENDS>
```

5. LISP-Map-Server Module

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

5.1. Module Structure

```

module: ietf-lisp-mapserver
augment /rt:routing/rt:control-plane-protocols
  /rt:control-plane-protocol/lisp:lisp:
    +--rw map-server!
      +--rw sites
        +--rw site* [site-id]
          +--rw site-id                uint64
          +--rw authentication-keys
            +--rw authentication-key* [auth-key-id]
              +--rw auth-key-id        string
              +--rw auth-algorithm-id*
                | lisp:auth-algorithm-type
              +--rw auth-key-value?    string
          +--rw xtr-ids* [xtr-id]
            +--rw xtr-id                uint64
            +--rw authentication-keys
              +--rw authentication-key* [auth-key-id]
                +--rw auth-key-id        string
                +--rw auth-algorithm-id*
                  | lisp:auth-algorithm-type
                +--rw auth-key-value?    string
      +--rw vpns
        +--rw vpn* [instance-id]
          +--rw instance-id            lisp-at:instance-id-type
          +--rw mappings
            +--rw mapping* [eid-id]
              +--rw eid-id              lisp:eid-id
              +--rw eid-address
                +--rw address-type
                  | lisp-address-family-ref
                +--rw (address)?
                  +--:(no-address)
                  | +--rw no-address?          empty
                  +--:(ipv4)
                  | +--rw ipv4?
                  |   inet:ipv4-address
                  +--:(ipv4-prefix)
                  | +--rw ipv4-prefix?
                  |   inet:ipv4-prefix
                  +--:(ipv6)
                  | +--rw ipv6?
                  |   inet:ipv6-address
                  +--:(ipv6-prefix)
                  | +--rw ipv6-prefix?
                  |   inet:ipv6-prefix
                  +--:(mac)
                  | +--rw mac?
                  |   yang:mac-address

```

```

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

```

```

+---:(nat-traversal)
|   +---rw nat-traversal
|       +---rw ms-udp-port?          uint16
|       +---rw etr-udp-port?        uint16
|       +---rw global-etr-rloc?
|           |
|           |   simple-address
|       +---rw ms-rloc?
|           |
|           |   simple-address
|       +---rw private-etr-rloc?
|           |
|           |   simple-address
|       +---rw rtr-rlocs*
|           |
|           |   simple-address
+---:(explicit-locator-path)
|   +---rw explicit-locator-path
|       +---rw hop* [hop-id]
|           +---rw hop-id            string
|           +---rw address?          simple-address
|           +---rw lrs-bits?         bits
+---:(source-dest-key)
|   +---rw source-dest-key
|       +---rw source?              simple-address
|       +---rw dest?                simple-address
+---:(key-value-address)
|   +---rw key-value-address
|       +---rw key?                  simple-address
|       +---rw value?                simple-address
+---:(service-path)
|   +---rw service-path
|       +---rw service-path-id?
|           |
|           |   service-path-id-type
|       +---rw service-index?        uint8
+---rw site-id*                      uint64
+---rw more-specifics-accepted?      boolean
+---rw mapping-expiration-timeout?   int16
+---ro first-registration-time?
|   |
|   |   yang:date-and-time
+---ro last-registration-time?
|   |
|   |   yang:date-and-time
+---rw mapping-records
|   +---rw mapping-record* [xtr-id]
|       +---rw xtr-id
|           |
|           |   lisp:xtr-id-type
|       +---rw site-id?              uint64
+---rw eid
|   +---rw address-type
|       |
|       |   lisp-address-family-ref
|   +---rw (address)?
|       |
|       |   +---:(no-address)

```

```

+---rw no-address?
    empty
+---:(ipv4)
+---rw ipv4?
    inet:ipv4-address
+---:(ipv4-prefix)
+---rw ipv4-prefix?
    inet:ipv4-prefix
+---:(ipv6)
+---rw ipv6?
    inet:ipv6-address
+---:(ipv6-prefix)
+---rw ipv6-prefix?
    inet:ipv6-prefix
+---:(mac)
+---rw mac?
    yang:mac-address
+---:(distinguished-name)
+---rw distinguished-name?
    distinguished-name-type
+---:(as-number)
+---rw as-number?
    inet:as-number
+---:(null-address)
+---rw null-address
    +---rw address?    empty
+---:(afi-list)
+---rw afi-list
    +---rw address-list*
        simple-address
+---:(instance-id)
+---rw instance-id
    +---rw instance-id?
        |
        | instance-id-type
+---rw mask-length?    uint8
+---rw address?
    simple-address
+---:(as-number-lcaf)
+---rw as-number-lcaf
    +---rw as?          inet:as-number
    +---rw address?     simple-address
+---:(application-data)
+---rw application-data
    +---rw address?
        |
        | simple-address
+---rw protocol?      uint8
+---rw ip-tos?        int32
+---rw local-port-low?

```

```

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

```

```

+---:(source-dest-key)
|   +---rw source-dest-key
|       +---rw source?    simple-address
|       +---rw dest?      simple-address
+---:(key-value-address)
|   +---rw key-value-address
|       +---rw key?       simple-address
|       +---rw value?     simple-address
+---:(service-path)
|   +---rw service-path
|       +---rw service-path-id?
|           service-path-id-type
|       +---rw service-index?    uint8
+---rw time-to-live?             uint32
+---ro creation-time?
|   yang:date-and-time
+---rw authoritative?           bits
+---rw static?                  boolean
+---rw (locator-list)?
+---:(negative-mapping)
|   +---rw map-reply-action?
|       map-reply-action
+---:(positive-mapping)
+---rw rlocs
|   +---rw locator* [locator-id]
|       +---rw locator-id
|           string
|   +---rw locator-address
|       +---rw address-type
|           lisp-address-family-ref
|   +---rw (address)?
|       +---:(no-address)
|           +---rw no-address?
|               empty
|       +---:(ipv4)
|           +---rw ipv4?
|               inet:ipv4-address
|       +---:(ipv4-prefix)
|           +---rw ipv4-prefix?
|               inet:ipv4-prefix
|       +---:(ipv6)
|           +---rw ipv6?
|               inet:ipv6-address
|       +---:(ipv6-prefix)
|           +---rw ipv6-prefix?
|               inet:ipv6-prefix
|       +---:(mac)
|           +---rw mac?

```



```

|           yang:mac-address
+---:(distinguished-name)
|   +---rw distinguished-name?
|       distinguished-name-type
+---:(as-number)
|   +---rw as-number?
|       inet:as-number
+---:(null-address)
|   +---rw null-address
|       +---rw address?
|           empty
+---:(afi-list)
|   +---rw afi-list
|       +---rw address-list*
|           simple-address
+---:(instance-id)
|   +---rw instance-id
|       +---rw instance-id?
|           instance-id-type
|       +---rw mask-length?
|           uint8
|       +---rw address?
|           simple-address
+---:(as-number-lcaf)
|   +---rw as-number-lcaf
|       +---rw as?
|           inet:as-number
|       +---rw address?
|           simple-address
+---:(application-data)
|   +---rw application-data
|       +---rw address?
|           simple-address
|       +---rw protocol?
|           uint8
|       +---rw ip-tos?
|           int32
|       +---rw local-port-low?
|           inet:port-number
|       +---rw local-port-high?
|           inet:port-number
|       +---rw remote-port-low?
|           inet:port-number
|       +---rw remote-port-high?
|           inet:port-number
+---:(geo-coordinates)
|   +---rw geo-coordinates
|       +---rw latitude?

```

```

|         bits
+---rw latitude-degrees?
|         uint8
+---rw latitude-minutes?
|         uint8
+---rw latitude-seconds?
|         uint8
+---rw longitude?
|         bits
+---rw longitude-degrees?
|         uint16
+---rw longitude-minutes?
|         uint8
+---rw longitude-seconds?
|         uint8
+---rw altitude?
|         int32
+---rw address?
|         simple-address
+---:(nat-traversal)
+---rw nat-traversal
+---rw ms-udp-port?
|         uint16
+---rw etr-udp-port?
|         uint16
+---rw global-etr-rloc?
|         simple-address
+---rw ms-rloc?
|         simple-address
+---rw private-etr-rloc?
|         simple-address
+---rw rtr-rlocs*
|         simple-address
+---:(explicit-locator-path)
+---rw explicit-locator-path
+---rw hop* [hop-id]
|         string
+---rw address?
|         simple-address
+---rw lrs-bits?
|         bits
+---:(source-dest-key)
+---rw source-dest-key
+---rw source?
|         simple-address
+---rw dest?
|         simple-address

```

```

+---: (key-value-address)
+---rw key-value-address
+---rw key?
|       simple-address
+---rw value?
|       simple-address
+---: (service-path)
+---rw service-path
+---rw service-path-id?
|       service-path-id-type
+---rw service-index?
|       uint8
+---rw priority?
|       uint8
+---rw weight?
|       uint8
+---rw multicast-priority?
|       uint8
+---rw multicast-weight?
|       uint8
+---ro counters
+---ro map-registers-in?                yang:counter64
+---ro map-registers-in-auth-failed?    yang:counter64
+---ro map-notify-records-out?           yang:counter64
+---ro proxy-reply-records-out?          yang:counter64
+---ro map-requests-forwarded-out?      yang:counter64
+---rw mapping-system-type?    lisp:mapping-system-ref
+---ro summary
+---ro number-configured-sites?    uint32
+---ro number-registered-sites?    uint32
+---ro af-datum
+---ro af-data* [address-type]
+---ro address-type
|       lisp-at:lisp-address-family-ref
+---ro number-configured-eids?    uint32
+---ro number-registered-eids?    uint32
+---ro counters
+---ro map-registers-in?                yang:counter64
+---ro map-registers-in-auth-failed?    yang:counter64
+---ro map-notify-records-out?           yang:counter64
+---ro proxy-reply-records-out?          yang:counter64
+---ro map-requests-forwarded-out?      yang:counter64

```

5.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp-mapserver@2021-02-22.yang"
module ietf-lisp-mapserver {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver";

  prefix lisp-ms;

  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note
  import ietf-lisp {
    prefix lisp;
    reference "RFC XXXX: LISP YANG model";
  }
  import ietf-lisp-address-types {
    prefix lisp-at;
    reference "RFC XXXX: LISP YANG model";
  }
  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management
       (NMDA version)";
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/lisp/>
    WG List:  <mailto:lisp@ietf.org>

    Editor:   Vina Ermagan
              <mailto:ermagan@gmail.com>

    Editor:   Alberto Rodriguez-Natal
              <mailto:natal@cisco.com>

    Editor:   Reshad Rahman
              <mailto:reshad@yahoo.com>";
  description
    "This YANG module defines the generic parameters for a LISP
    Map-Server. The module can be extended by vendors to define
    vendor-specific parameters and policies."
```

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```
";

reference "RFC XXXX";

revision 2021-02-22 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6833";
}

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

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

    leaf map-registers-in {
      type yang:counter64;
      description "Number of incoming Map-Register messages";
    }

    leaf map-registers-in-auth-failed {
      type yang:counter64;
      description
        "Number of incoming Map-Register messages failed
        authentication";
    }

    leaf map-notify-records-out {
```

```
        type yang:counter64;
        description
            "Number of outgoing Map-Notify records";
    }

    leaf proxy-reply-records-out {
        type yang:counter64;
        description
            "Number of outgoing proxy Map-Reply records";
    }

    leaf map-requests-forwarded-out {
        type yang:counter64;
        description
            "Number of outgoing Map-Requests forwarded to ETR";
    }
}

augment "/rt:routing/rt:control-plane-protocols"
+ "/rt:control-plane-protocol/lisp:lisp" {
    when "lisp:lisp-role/lisp:lisp-role-type = 'lisp-ms:ms'" {
        description
            "Augment is valid when LISP device type is Map-Server.";
    }
    description
        "This augments the LISP devices list with Map-Server
        specific parameters.";
    container map-server {
        presence "LISP Map-Server operation enabled";
        description
            "Map-Server parameters.";
        container sites{
            description
                "Sites to accept registrations from.";
            list site {
                key site-id;
                description
                    "Site that can send registrations.";
                leaf site-id {
                    type uint64;
                    description "Site ID";
                }
            }
            uses lisp:auth-key;
            list xtr-ids {
                key xtr-id;
                description "xTR-ID specific configuration.";
                leaf xtr-id {
```

```
        type uint64;
        description "xTR ID";
    }
    uses lisp:auth-key;
}
}
}
container vpns {
    description
        "VPNs for which the Map-Server accepts registrations.";
    list vpn {
        key "instance-id";
        description
            "VPN instances in the Map-Server.";
        leaf instance-id {
            type lisp-at:instance-id-type;
            description
                "VPN identifier.";
        }
        container mappings {
            description
                "EIDs registered by device.";
            list mapping {
                key "eid-id";
                description
                    "List of EIDs registered by device.";
                leaf eid-id {
                    type lisp:eid-id;
                    description
                        "Id of the EID registered.";
                }
            }
            container eid-address {
                uses lisp-at:lisp-address;
                description
                    "EID in generic LISP address format registered
                    with the Map-Server.";
            }
            leaf-list site-id {
                type uint64;
                description "Site ID";
            }
            leaf more-specifics-accepted {
                type boolean;
                default "false";
                description
                    "Flag indicating if more specific prefixes
                    can be registered.";
            }
        }
    }
}
```

```
    leaf mapping-expiration-timeout {
      type int16;
      units "seconds";
      default "180"; //3 times the mapregister int
      description
        "Time before mapping is expired if no new
         registrations are received.";
    }
    leaf first-registration-time {
      type yang:date-and-time;
      config false;
      description
        "Time at which the first registration for this
         EID was received";
    }
    leaf last-registration-time {
      type yang:date-and-time;
      config false;
      description
        "Time at which the last registration for this EID
         was received";
    }
    container mapping-records {
      description
        "Datastore of registered mappings.";
      list mapping-record {
        key xtr-id;
        description
          "Registered mapping.";
        leaf xtr-id {
          type lisp:xtr-id-type;
          description "xTR ID";
        }
        leaf site-id {
          type uint64;
          description "Site ID";
        }
        uses lisp:mapping;
      }
    }
  }
  uses ms-counters;
}

leaf mapping-system-type {
  type lisp:mapping-system-ref;
  description
```



```
        "A reference to the mapping system";
    }

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

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

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

6. LISP-Map-Resolver Module

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

6.1. Module Structure

```
module: ietf-lisp-mapresolver
  augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol/lisp:lisp:
    +--rw map-resolver!
      +--rw mapping-system-type?    lisp:mapping-system-ref
      +--rw ms-address?             inet:ip-address
```

6.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp-mapresolver@2019-02-23.yang"
module ietf-lisp-mapresolver {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-mapresolver";

  prefix lisp-mr;

  // RFC Ed.: replace occurrences of XXXX with actual RFC number
  // and remove this note
  import ietf-lisp {
    prefix lisp;
    reference "RFC XXXX: LISP YANG model";
  }
  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management
       (NMDA version)";
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/lisp/>
     WG List: <mailto:lisp@ietf.org>

     Editor:  Vina Ermagan
              <mailto:ermagan@gmail.com>

     Editor:  Alberto Rodriguez-Natal
              <mailto:natal@cisco.com>
```

```
Editor:   Reshad Rahman
         <mailto:reshad@yahoo.com>;

description
  "This YANG module defines the generic parameters for a LISP
  Map-Resolver. The module can be extended by vendors to define
  vendor-specific parameters and policies.

  Copyright (c) 2018 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

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

reference "RFC XXXX";

revision 2019-02-23 {
  description
    "Initial revision.";
  reference
    "https://tools.ietf.org/html/rfc6833";
}
identity mr {
  base lisp:lisp-role;
  description
    "LISP Map-Resolver.";
}

augment "/rt:routing/rt:control-plane-protocols"
  + "/rt:control-plane-protocol/lisp:lisp" {
  when "lisp:lisp-role/lisp:lisp-role-type = 'lisp-mr'" {
    description
      "Augment is valid when LISP device type is Map-Resolver.";
  }
  description
    "This augments the LISP devices list with Map-Resolver
    specific parameters.";
  container map-resolver {
    presence "LISP Map-Resolver operation enabled";
    description
      "Map-Resolver parameters.";
  }
}
```

```
    leaf mapping-system-type {
      type lisp:mapping-system-ref;
      description
        "A reference to the mapping system";
    }
    leaf ms-address {
      when "../mapping-system-type="
        + "'lisp:single-node-mapping-system'";
      type inet:ip-address;
      description
        "address to reach the Map Server when "
        + "lisp-mr:single-node-mapping-system is being used.";
    }
  }
}
}
<CODE ENDS>
```

7. LISP-Address-Types Module

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

7.1. Module Definition

```
<CODE BEGINS> file "ietf-lisp-address-types@2021-02-22.yang"
module ietf-lisp-address-types {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types";

  prefix lisp-at;

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types";
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/lisp/>
    WG List:  <mailto:lisp@ietf.org>
```

Editor: Vina Ermagan
<mailto:ermagan@gmail.com>

Editor: Alberto Rodriguez-Natal
<mailto:natal@cisco.com>

Editor: Reshad Rahman
<mailto:reshad@yahoo.com>;

description

"This YANG module defines the LISP Canonical Address Formats (LCAF) for LISP. The module can be extended by vendors to define vendor-specific parameters.

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```
";
// RFC Ed.: replace XXXX with actual RFC number and remove
// this note
reference "RFC XXXX";

revision 2021-02-22 {
  description
    "Initial revision.";
  reference
    "RFC8060: LISP Canonical Address Format (LCAF)";
}
identity lisp-address-family {
  description
    "Base identity from which identities describing LISP address
    families are derived.";
}
identity no-address-afi {
  base lisp-address-family;
  description
    "IANA Reserved.";
}
identity ipv4-afi {
```

```
    base lisp-address-family;
    description
      "IANA IPv4 address family.";
  }
  identity ipv4-prefix-afi {
    base lisp-address-family;
    description
      "IANA IPv4 address family prefix.";
  }
  identity ipv6-afi {
    base lisp-address-family;
    description
      "IANA IPv6 address family.";
  }
  identity ipv6-prefix-afi {
    base lisp-address-family;
    description
      "IANA IPv6 address family prefix.";
  }
  identity mac-afi {
    base lisp-address-family;
    description
      "IANA MAC address family.";
  }
  identity distinguished-name-afi {
    base lisp-address-family;
    description
      "IANA Distinguished Name address family.";
  }
  identity as-number-afi {
    base lisp-address-family;
    description
      "IANA AS Number address family.";
  }
  identity lcaf {
    base lisp-address-family;
    description
      "IANA LISP Canonical Address Format address family.";
  }
  identity null-address-lcaf {
    base lcaf;
    description
      "Null body LCAF type.";
  }
  identity afi-list-lcaf {
    base lcaf;
    description
      "AFI-List LCAF type.";
```

```
}
identity instance-id-lcaf {
  base lcaf;
  description
    "Instance-ID LCAF type.";
}
identity as-number-lcaf {
  base lcaf;
  description
    "AS Number LCAF type.";
}
identity application-data-lcaf {
  base lcaf;
  description
    "Application Data LCAF type.";
}
identity geo-coordinates-lcaf {
  base lcaf;
  description
    "Geo-coordinates LCAF type.";
}
identity opaque-key-lcaf {
  base lcaf;
  description
    "Opaque Key LCAF type.";
}
identity nat-traversal-lcaf {
  base lcaf;
  description
    "NAT-Traversal LCAF type.";
}
identity nonce-locator-lcaf {
  base lcaf;
  description
    "Nonce-Locator LCAF type.";
}
identity multicast-info-lcaf {
  base lcaf;
  description
    "Multicast Info LCAF type.";
}
identity explicit-locator-path-lcaf {
  base lcaf;
  description
    "Explicit Locator Path LCAF type.";
}
identity security-key-lcaf {
  base lcaf;
```

```
    description
      "Security Key LCAF type.";
  }
  identity source-dest-key-lcaf {
    base lcaf;
    description
      "Source/Dest LCAF type.";
  }
  identity replication-list-lcaf {
    base lcaf;
    description
      "Replication-List LCAF type.";
  }
  identity json-data-model-lcaf {
    base lcaf;
    description
      "JSON Data Model LCAF type.";
  }
  identity key-value-address-lcaf {
    base lcaf;
    description
      "Key/Value Address LCAF type.";
  }
  identity encapsulation-format-lcaf {
    base lcaf;
    description
      "Encapsulation Format LCAF type.";
  }
  identity service-path-lcaf {
    base lcaf;
    description
      "Service Path LCAF type.";
  }
  typedef instance-id-type {
    type uint32 {
      range "0..16777215";
    }
    description
      "Defines the range of values for an Instance ID.";
  }
  typedef service-path-id-type {
    type uint32 {
      range "0..16777215";
    }
    description
      "Defines the range of values for a Service Path ID.";
  }
  typedef distinguished-name-type {
```



```
    type string;
    description
      "Distinguished Name address.";
    reference
      "http://www.iana.org/assignments/address-family-numbers/
      address-family-numbers.xhtml";
  }
  typedef simple-address {
    type union {
      type inet:ip-address;
      type inet:ip-prefix;
      type yang:mac-address;
      type distinguished-name-type;
      type inet:as-number;
    }
    description
      "Union of address types that can be part of LCAFs.";
  }
  typedef lisp-address-family-ref {
    type identityref {
      base lisp-address-family;
    }
    description
      "LISP address family reference.";
  }
  typedef lcaf-ref {
    type identityref {
      base lcaf;
    }
    description
      "LCAF types reference.";
  }

  grouping lisp-address {
    description
      "Generic LISP address.";
    leaf address-type {
      type lisp-address-family-ref;
      mandatory true;
      description
        "Type of the LISP address.";
    }
    choice address {
      description
        "Various LISP address types, including IP, MAC, and LCAF.";

      leaf no-address {
        when "../address-type = 'lisp-at:no-address-afi'" {
```

```
        description
            "When AFI is 0.";
    }
    type empty;
    description
        "No address.";
}
leaf ipv4 {
    when "../address-type = 'lisp-at:ipv4-afi'" {
        description
            "When AFI is IPv4.";
    }
    type inet:ipv4-address;
    description
        "IPv4 address.";
}
leaf ipv4-prefix {
    when "../address-type = 'lisp-at:ipv4-prefix-afi'" {
        description
            "When AFI is IPv4.";
    }
    type inet:ipv4-prefix;
    description
        "IPv4 prefix.";
}
leaf ipv6 {
    when "../address-type = 'lisp-at:ipv6-afi'" {
        description
            "When AFI is IPv6.";
    }
    type inet:ipv6-address;
    description
        "IPv6 address.";
}
leaf ipv6-prefix {
    when "../address-type = 'lisp-at:ipv6-prefix-afi'" {
        description
            "When AFI is IPv6.";
    }
    type inet:ipv6-prefix;
    description
        "IPv6 address.";
}
leaf mac {
    when "../address-type = 'lisp-at:mac-afi'" {
        description
            "When AFI is MAC.";
    }
}
```

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

```
        "List of AFI addresses.";
    }
}
container instance-id {
    when "../address-type = 'lisp-at:instance-id-lcaf'" {
        description
            "When LCAF type is Instance ID as per RFC8060.";
    }
    description
        "Instance ID LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
        #section-4.2";
    leaf instance-id {
        type instance-id-type;
        description
            "Instance ID value.";
    }
    leaf mask-length {
        type uint8;
        description
            "Mask length.";
    }
    leaf address {
        type simple-address;
        description
            "AFI address.";
    }
}
container as-number-lcaf {
    when "../address-type = 'lisp-at:as-number-lcaf'" {
        description
            "When LCAF type is AS-Number.";
    }
    description
        "AS Number LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
        #section-4.3";
    leaf as {
        type inet:as-number;
        description
            "AS number.";
    }
    leaf address {
        type simple-address;
        description
            "AFI address.";
```

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

```
container geo-coordinates {
  when "../address-type = 'lisp-at:geo-coordinates-lcaf'" {
    description
      "When LCAF type is Geo-coordinates.";
  }
  description
    "Geo-coordinates LCAF type. Coordinates are specified
    using the WGS 84 (World Geodetic System 1984) reference
    coordinate system";
  reference
    "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
    #section-4.5";
  leaf latitude {
    type bits {
      bit N {
        description
          "Latitude bit.";
      }
    }
    description
      "Bit that selects between North and South latitude.";
  }
  leaf latitude-degrees {
    type uint8 {
      range "0 .. 90";
    }
    description
      "Degrees of latitude.";
  }
  leaf latitude-minutes {
    type uint8 {
      range "0..59";
    }
    description
      "Minutes of latitude.";
  }
  leaf latitude-seconds {
    type uint8 {
      range "0..59";
    }
    description
      "Seconds of latitude.";
  }
  leaf longitude {
    type bits {
      bit E {
        description
          "Longitude bit.";
      }
    }
  }
}
```

```
    }
  }
  description
    "Bit that selects between East and West longitude.";
}
leaf longitude-degrees {
  type uint16 {
    range "0 .. 180";
  }
  description
    "Degrees of longitude.";
}
leaf longitude-minutes {
  type uint8 {
    range "0..59";
  }
  description
    "Minutes of longitude.";
}
leaf longitude-seconds {
  type uint8 {
    range "0..59";
  }
  description
    "Seconds of longitude.";
}
leaf altitude {
  type int32;
  description
    "Height relative to sea level in meters.";
}
leaf address {
  type simple-address;
  description
    "AFI address.";
}
}
container nat-traversal {
  when "../address-type = 'lisp-at:nat-traversal-lcaf'" {
    description
      "When LCAF type is NAT-Traversal.";
  }
  description
    "NAT-Traversal LCAF type.";
  reference
    "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10  
#section-4.6";
  leaf ms-udp-port {
```

```
        type uint16;
        description
            "Map-Server UDP port (set to 4342).";
    }
    leaf etr-udp-port {
        type uint16;
        description
            "ETR UDP port.";
    }
    leaf global-etr-rloc {
        type simple-address;
        description
            "Global ETR RLOC address.";
    }
    leaf ms-rloc {
        type simple-address;
        description
            "Map-Server RLOC address.";
    }
    leaf private-etr-rloc {
        type simple-address;
        description
            "Private ETR RLOC address.";
    }
    leaf-list rtr-rlocs {
        type simple-address;
        description
            "List of RTR RLOC addresses.";
    }
}

container explicit-locator-path {
    when "../address-type = 'lisp-at:explicit-locator-path-lcaf'" {
        description
            "When LCAF type type is Explicit Locator Path.";
    }
    description
        "Explicit Locator Path LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
        #section-4.9";
    list hop {
        key "hop-id";
        ordered-by user;
        description
            "List of locator hops forming the explicit path.";
        leaf hop-id {
            type string {
                length "1..64";
            }
        }
    }
}
```



```
        pattern '[a-zA-Z0-9\-\_\.]*';
    }
    description
        "Unique identifier for the hop.";
}
leaf address {
    type simple-address;
    description
        "AFI address.";
}
leaf lrs-bits {
    type bits{
        bit lookup {
            description
                "Lookup bit.";
        }
        bit rloc-probe {
            description
                "RLOC-probe bit.";
        }
        bit strict {
            description
                "Strict bit.";
        }
    }
    description
        "Flag bits per hop.";
}
}
}
container source-dest-key {
    when "../address-type = 'lisp-at:source-dest-key-lcaf'" {
        description
            "When LCAF type type is Source/Dest.";
    }
    description
        "Source/Dest LCAF type.";
    reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10#section-4.11";
    leaf source {
        type simple-address;
        description
            "Source address.";
    }
    leaf dest {
        type simple-address;
        description
```

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

7.2. Data Model examples

This section presents some simple and illustrative examples on how to configure LISP.

7.2.1. LISP protocol instance

The following is an example configuration for a LISP protocol instance with the name "LISP1". There are also 2 VNIs configured.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <network-instances
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
    <network-instance>
      <name>VRF-BLUE</name>
      <vrf-root/>
      <enabled>true</enabled>
    </network-instance>
    <network-instance>
      <name>VRF-RED</name>
      <vrf-root/>
      <enabled>true</enabled>
    </network-instance>
  </network-instances>
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
          lisp:lisp
        </type>
        <name>LISP1</name>
        <lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
          <lisp-role>
            <lisp-role-type>etr</lisp-role-type>
          </lisp-role>
          <lisp-role>
            <lisp-role-type>itr</lisp-role-type>
          </lisp-role>
          <vpns>
            <vpn>
              <instance-id>1000</instance-id>
              <iid-name>VRF-BLUE</iid-name>
            </vpn>
            <vpn>
              <instance-id>2000</instance-id>
              <iid-name>VRF-RED</iid-name>
            </vpn>
          </vpns>
        </lisp>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>
```

7.2.2. LISP ITR

The following is an example configuration for ITR functionality under "LISP1". There are 2 Map-Resolvers configured.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
          lisp:lisp
        </type>
        <name>LISP1</name>
        <lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
          <lisp-role>
            <lisp-role-type>itr</lisp-role-type>
          </lisp-role>
          <itr xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp-itr">
            <map-resolvers>
              <map-resolver>2001:db8:203:0:113::1</map-resolver>
              <map-resolver>2001:db8:204:0:113::1</map-resolver>
            </map-resolvers>
          </itr>
        </lisp>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>
```

7.2.3. LISP ETR

The following is an example configuration for ETR functionality under "LISP1". There are 2 Map-Servers and 2 local EIDs configured.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <network-instances
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
    <network-instance>
      <name>VRF-BLUE</name>
      <vrf-root/>
      <enabled>true</enabled>
    </network-instance>
  </network-instances>
</config>
```

```
<network-instance>
  <name>VRF-RED</name>
  <vrf-root/>
  <enabled>true</enabled>
</network-instance>
</network-instances>
<routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
  <control-plane-protocols>
    <control-plane-protocol>
      <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
        lisp:lisp
      </type>
      <name>LISP1</name>
      <lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
        <lisp-role>
          <lisp-role-type>etr</lisp-role-type>
        </lisp-role>
        <lisp-router-id>
          <site-id>1</site-id>
        </lisp-router-id>
        <vpns>
          <vpn>
            <instance-id>1000</instance-id>
            <iid-name>VRF-BLUE</iid-name>
          </vpn>
          <vpn>
            <instance-id>2000</instance-id>
            <iid-name>VRF-RED</iid-name>
          </vpn>
        </vpns>
        <etr xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp-etr">
          <map-servers>
            <map-server>
              <ms-address>2001:db8:203:0:113::1</ms-address>
              <authentication-keys>
                <authentication-key>
                  <auth-key-id>key1</auth-key-id>
                  <auth-algorithm-id>
                    hmac-sha-256-128
                  </auth-algorithm-id>
                  <auth-key-value>*Kye^$$1#gb91U04zpa</auth-key-value>
                </authentication-key>
              </authentication-keys>
            </map-server>
            <map-server>
              <ms-address>2001:db8:204:0:113::1</ms-address>
              <authentication-keys>
                <authentication-key>
```

```
        <auth-key-id>key1</auth-key-id>
        <auth-algorithm-id>
            hmac-sha-256-128
        </auth-algorithm-id>
        <auth-key-value>*Kye^$$1#gb91U04zpa</auth-key-value>
    </authentication-key>
</authentication-keys>
</map-server>
</map-servers>
<local-eids>
    <vpn>
        <instance-id>1000</instance-id>
        <eids>
            <local-eid>
                <id>2001:db8:400:0:100::0</id>
                <eid-address>
                    <address-type xmlns:lisp-at=
                        "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
                        lisp-at:ipv6-prefix-afi
                    </address-type>
                    <ipv6-prefix>2001:db8:400:0:100::/80</ipv6-prefix>
                </eid-address>
            </local-eid>
        </eids>
    </vpn>
    <vpn>
        <instance-id>2000</instance-id>
        <eids>
            <local-eid>
                <id>2001:db8:800:0:200::0</id>
                <eid-address>
                    <address-type xmlns:lisp-at=
                        "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
                        lisp-at:ipv6-prefix-afi
                    </address-type>
                    <ipv6-prefix>2001:db8:800:0:200::/80</ipv6-prefix>
                </eid-address>
            </local-eid>
        </eids>
    </vpn>
</local-eids>
</etr>
</lisp>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</config>
```

7.2.4. LISP Map-Server

The following is an example configuration for Map-Server functionality under "LISP1". There are 2 mappings configured.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:lisp="urn:ietf:params:xml:ns:yang:ietf-lisp">
          lisp:lisp
        </type>
        <name>LISP1</name>
        <lisp xmlns="urn:ietf:params:xml:ns:yang:ietf-lisp">
          <lisp-role>
            <lisp-role-type xmlns:lisp-ms=
              "urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver">
              lisp-ms:ms
            </lisp-role-type>
          </lisp-role>
          <map-server xmlns=
            "urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver">
            <sites>
              <site>
                <site-id>1</site-id>
                <authentication-keys>
                  <authentication-key>
                    <auth-key-id>key1</auth-key-id>
                    <auth-algorithm-id>
                      hmac-sha-256-128
                    </auth-algorithm-id>
                    <auth-key-value>*Kye^$$1#gb91U04zpa</auth-key-value>
                  </authentication-key>
                </authentication-keys>
              </site>
            </sites>
            <vpns>
              <vpn>
                <instance-id>1000</instance-id>
                <mappings>
                  <mapping>
                    <eid-id>1</eid-id>
                    <eid-address>
                      <address-type xmlns:lisp-at=
                        "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
                        lisp-at:ipv6-prefix-afi
```



```

        </address-type>
        <ipv6-prefix>2001:db8:400:0:100::/80</ipv6-prefix>
      </eid-address>
    </mapping>
  </mappings>
</vpn>
<vpn>
  <instance-id>2000</instance-id>
  <mappings>
    <mapping>
      <eid-id>1</eid-id>
      <eid-address>
        <address-type xmlns:lisp-at=
"urn:ietf:params:xml:ns:yang:ietf-lisp-address-types">
          lisp-at:ipv6-prefix-afi
        </address-type>
        <ipv6-prefix>2001:db8:800:0:200::/80</ipv6-prefix>
      </eid-address>
    </mapping>
  </mappings>
</vpn>
</vpns>
</map-server>
</lisp>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</config>

```

8. Acknowledgments

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

9. IANA Considerations

The IANA is requested to assign a new namespace URI from the IETF XML registry.

This document registers the following namespace URIs in the IETF XML registry [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-lisp

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-itr

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-etr

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-mapresolver

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-lisp-address-types

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

This document registers the following YANG modules in the "YANG Module Names" registry [RFC6020]:

Name: ietf-lisp

Namespace: urn:ietf:params:xml:ns:yang:ietf-lisp

Prefix: lisp

Reference: RFC XXX

Name: ietf-lisp-itr

Namespace: urn:ietf:params:xml:ns:yang:ietf-lisp-itr

Prefix: lisp-itr

Reference: RFC XXX

Name: ietf-lisp-etr

Namespace: urn:ietf:params:xml:ns:yang:ietf-lisp-etr

Prefix: lisp-etr

Reference: RFC XXX

Name: ietf-lisp-mapserver

Namespace: urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver

Prefix: lisp-ms

Reference: RFC XXX

Name: ietf-lisp-mapresolver

Namespace: urn:ietf:params:xml:ns:yang:ietf-lisp-mapresolver

Prefix: lisp-mr

Reference: RFC XXX

Name: ietf-lisp-address-types

Namespace: urn:ietf:params:xml:ns:yang:ietf-lisp-address-types

Prefix: lisp-at

Reference: RFC XXX

10. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS

[RFC8446].

The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a pre-configured subset of all available NETCONF or RESTCONF protocol operations and content.

The security considerations of LISP control-plane [RFC6833] and LISP data-plane [RFC6830] as well as the LISP threat analysis [RFC7835] apply to this YANG model.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/  
lisp:lisp/
```

Access to the locator-sets node may modify which interfaces are used for data and/or control traffic as well as affect the load balancing of data-plane traffic. Access to the lisp-role node may prevent the device from performing its intended data-plane and/or control-plane operation. Access to the router-id node allows to modify the unique identifier of the device, which may result in disruption of its LISP control-plane operation. Access to the vpn node may allow to redirect data-plane traffic to erroneous local or remote network instances.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:map-server
```

Access to the sites node can prevent authorized devices from registering mappings in the Map-Server and/or allow unauthorized devices to do so. Access to the vpn node can result in corrupted mapping state that may propagate across the LISP network, potentially resulting in forwarding of data-plane traffic to arbitrary destinations and general disruption of the data-plane operation. Access to mapping-system-type and/or ddt-mapping-system nodes may prevent the device from connecting to the Mapping System infrastructure and consequentially to attract Map-Request messages.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:map-resolver
```

Access to mapping-system-type, ms-address and/or ddt-mapping-system nodes may prevent the device to connect to the Mapping System infrastructure and forward Map-Request messages.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp/lisp:itr
```

Access to the rloc-probing node can increase the control-plane overhead in the device or affect the capability of the device to detect failures on the underlay. Access to the itr-rlocs node may prevent the device from getting Map-Reply messages. Access to the map-resolvers node can prevent the device from sending its Map-Request messages to valid Map-Resolvers. Access to the proxy-etr nodes can affect the capability of the device to send data-plane traffic towards non-LISP destinations. Access to the map-cache node can result in forwarding of data-plane traffic to arbitrary destinations and general disruption of data-plane operation.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp/lisp:etr
```

Access to the map-servers node can prevent the device from registering its local mappings into the Mapping System. Access to the local-eids node can disrupt data-plane operation on the device and/or result in the device registering corrupted mappings into the Mapping System.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp
```

Access to the locator-sets node can expose the locators the device is using for its control and/or data operation. Access to the lisp-role node can disclose the LISP roles instantiated at the device which facilitates mounting attacks against the device. Access to the router-id node can expose the unique identifier of device which may allow a third party to track its control-plane operation and/or impersonate the device. Access to the vpn node can leak the local mapping between LISP Instance IDs and local network instances.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp/lisp:map-server
```

Access to the sites node can expose the credentials used to register mappings and allow unauthorized devices to do so. Access to the vpn node can expose the mappings currently registered in the device, which has privacy implications. Access to the mapping-system-type node may reveal the Mapping System in use which can be used to mount attacks against the device and/or the Mapping System. Access to the summary and counters nodes may expose operational statistics of the device.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:map-resolver
```

Access to the mapping-system-type node may reveal the Mapping System in use which can be used to mount attacks against the device and/or the Mapping System. Access to the ms-address and/or ddt-mapping-system nodes can leak the information about the Mapping System infrastructure used by the device, which can be used to block communication and/or mount attacks against it.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:itr
```

Access to the rloc-probing node can expose if and how the device is using control-plane signaling to probe underlay locators. Access to the itr-rlocs node may disclose the addresses the device is using to receive Map-Reply messages. Access to the map-resolvers node can expose the Map-Resolvers used by the device, which can be used to mount attacks against the device and/or the Mapping System. Access to the proxy-etrns node can disclose the PETRs used by the device, which can be used to mount attacks against the device and/or PETRs. Access to the map-cache node can expose the mappings currently cached in the device, which has privacy implications.

```
/rt:routing/rt:control-plane-protocols/rt:control-plane-  
protocol/lisp:lisp/lisp:etr
```

Access to the map-servers node can expose the credentials used by the device to register mappings into the Mapping System allowing an unauthorized device to impersonate and register mappings on behalf the authorized device. Access to the local-eids node can expose the local EIDs currently being served by the device, which has privacy implications.

11. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <<https://www.rfc-editor.org/info/rfc6832>>.
- [RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, DOI 10.17487/RFC6833, January 2013, <<https://www.rfc-editor.org/info/rfc6833>>.
- [RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, DOI 10.17487/RFC6836, January 2013, <<https://www.rfc-editor.org/info/rfc6836>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.

- [RFC7835] Saucez, D., Iannone, L., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Threat Analysis", RFC 7835, DOI 10.17487/RFC7835, April 2016, <<https://www.rfc-editor.org/info/rfc7835>>.
- [RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, DOI 10.17487/RFC8022, November 2016, <<https://www.rfc-editor.org/info/rfc8022>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.
- [RFC8111] Fuller, V., Lewis, D., Ermagan, V., Jain, A., and A. Smirnov, "Locator/ID Separation Protocol Delegated Database Tree (LISP-DDT)", RFC 8111, DOI 10.17487/RFC8111, May 2017, <<https://www.rfc-editor.org/info/rfc8111>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

[RFC8529] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Data Model for Network Instances", RFC 8529, DOI 10.17487/RFC8529, March 2019, <<https://www.rfc-editor.org/info/rfc8529>>.

Authors' Addresses

Vina Ermagan
Google
United States of America
Email: ermagan@gmail.com

Alberto Rodriguez-Natal
Cisco Systems
San Jose, CA
United States of America
Email: natal@cisco.com

Florin Coras
Cisco Systems
San Jose, CA
United States of America
Email: fcoras@cisco.com

Carl Moberg
Avassa
Email: calle@avassa.io

Reshad Rahman
Canada
Email: reshad@yahoo.com

Albert Cabellos-Aparicio
Technical University of Catalonia
Barcelona
Spain
Email: acabello@ac.upc.edu

Fabio Maino
Cisco Systems
San Jose, CA
United States of America

Email: fmaino@cisco.com

LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: January 3, 2019

A. Rodriguez-Natal
V. Ermagan
F. Maino
D. Dukes
P. Camarillo
C. Filsfils
Cisco Systems, Inc.
July 2, 2018

LISP Control Plane for SRv6 Endpoint Mobility
draft-rodrigueznatal-lisp-srv6-00

Abstract

This document describes the use of the LISP Control Plane to support endpoint mobility and Location/ID separation in Segment Routing v6 (SRv6) deployments.

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 January 3, 2019.

Copyright Notice

Copyright (c) 2018 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. Overview	3
3. Operation	4
3.1. Provisioning	4
3.2. Endpoint Registration	5
3.3. Endpoint Resolution	6
3.4. SR Policy Resolution	6
3.4.1. Decentralized	7
3.4.2. Centralized	7
3.5. Traffic Encapsulation	8
3.6. Endpoint Mobility	9
4. Segment Routing LCAF (SR LCAF)	9
4.1. SR-MPLS	10
4.2. SRv6	11
4.3. Traffic Steering Tag	12
5. References	12
5.1. Normative References	12
5.2. Informative References	13
Authors' Addresses	13

1. Introduction

This document defines how to use the LISP Control Plane [I-D.ietf-lisp-rfc6833bis] to support endpoint mobility and Location/ID separation in those IPv6 deployments that are using SRv6 Network Programming [I-D.filsfils-spring-srv6-network-programming]. The LISP control plane is used to lookup "where" an endpoint is located, while SRv6 specifies "how" to program the SRv6 network infrastructure to transport traffic to that location.

To enable this, the egress Provider Edge (PE) nodes register via LISP control plane their local SRv6 Segment IDs (SIDs) to be used to reach endpoints attached to them. Ingress PE nodes retrieve via LISP control plane this mapping information, and use SRv6 network programming to encapsulate and steer the traffic towards the destination egress PE node.

The LISP control plane provides on-demand endpoint-to-SID mapping, as well as support for anchorless dynamic endpoint mobility.

2. Overview

The ingress PE nodes receive traffic from endpoints in their local networks, encapsulate it in IPv6 packets following SRv6 policies and forward it to the SRv6 domain. Similarly, egress PE nodes receive SRv6 traffic from the SRv6 domain, remove the SRv6 encapsulation and forward the traffic to one of their locally attached networks according to the decapsulation SID received in the packets. Ingress PE nodes and egress PE nodes can be co-located in the same node, in this document we use "PE node" to refer to those collocated nodes.

This specification leverages the LISP Mapping System to store mappings of endpoints to decapsulation SIDs. Endpoints are neither LISP nor SRv6 aware and can roam freely across different PE nodes. The mappings in the LISP Mapping System can be used by PE nodes to know to which PE node an endpoint is currently connected. The LISP Mapping System is composed of LISP Map-Resolvers and Map-Servers but, for convenience, this document refers to the Mapping System as a single logical entity.

To use the LISP Mapping System, in this specification the PE nodes also implement the control-plane logic of LISP Ingress/Egress Tunnel Routers (xTRs). As a LISP xTR, a PE node keeps a local database of all the endpoints that are attached to its local network(s). It also keeps a list of local decapsulation function SIDs that map to each of its local network(s). A PE node registers into the LISP Mapping System its local endpoint-to-SID mappings. These mappings also contain the traffic steering tag associated with the endpoint. In addition to its local mappings, a PE node also keeps a local map-cache of remote endpoint-to-SID mappings that it uses to find the SID to use to encapsulate traffic towards a remote endpoint.

This specification also assumes an SR Path Computation Element (PCE) [RFC4655] that can compute and provide SRv6 paths from a given ingress PE node to a given egress PE node. The paths are based on the ingress and egress PE nodes and on the traffic steering tag that is associated with the destination endpoint. Figure 1 shows an example diagram to be used as a reference for the architecture described in this document.

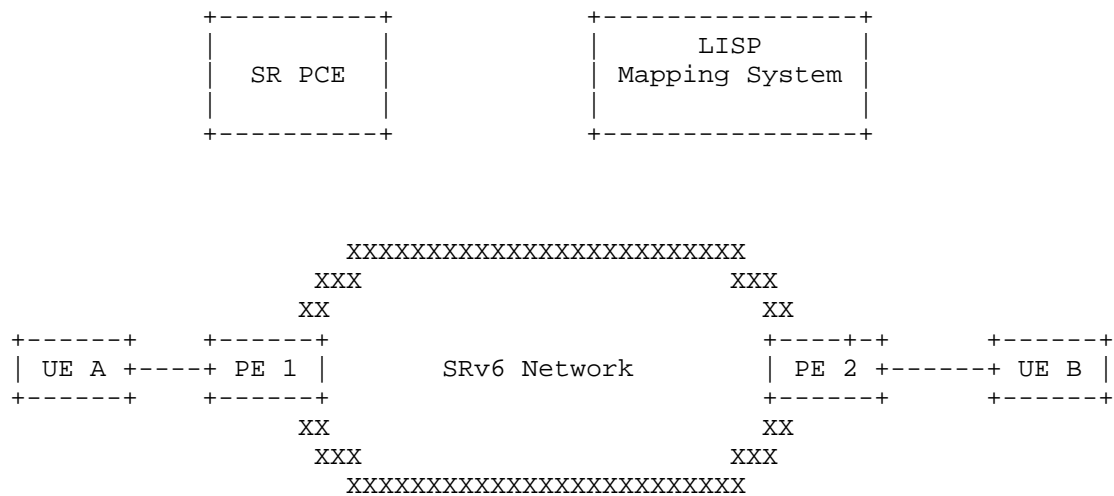


Figure 1: Reference Architecture

3. Operation

3.1. Provisioning

Each PE node is connected to the SRv6 domain and to one or more local networks. These local networks may represent different tenants/VPNs. Each tenant network is globally identified via a unique Instance Identifier (IID). Locally, these tenant networks are identified at each PE node by the local IP table assigned to them. While the IID for a tenant network is global across the domain, the local IP table assigned to it in a given PE node is local to that PE node. Each PE node needs to be provisioned with the one-to-one mapping of global IID to local IP table per each tenant network it is serving. The provisioning of the IID to the local IP table is out of the scope of this document. PE nodes use this IID to local IP table information to know which IID they need to use when requesting mappings for traffic coming from a particular tenant network (i.e. from a particular local IP table).

Per each local IP table, each PE node has a local SRv6 "End.DT46" function (see [I-D.filsfils-spring-srv6-network-programming]) that decapsulates SRv6 traffic and forwards the inner traffic for lookup into that particular IP table. The PE node concatenates one of its local SRv6 locators with each of these decapsulation functions to create SIDs that point to each of the different tenant networks it is serving. These SIDs are kept onto the "My local SIDs table" of the

PE node and they are used to decapsulate incoming SRv6 traffic into the appropriate local IP table.

Beyond SRv6 state, each PE node has to be provisioned with the address of at least one Map-Resolver it will use to retrieve remote endpoint-to-SID mappings from the Mapping System. Similarly, it has to be provisioned with the address of the Map-Server to which it is going to register their local endpoint-to-SID mappings.

3.2. Endpoint Registration

Endpoints attach to the local networks served by PE nodes. Upon attachment of an endpoint, the PE node will correlate the local IP table that corresponds to the local network where the endpoint attached to a global IID. It does so by using its local information of global IID to local IP table. The local IP table also correlates with the local SID that remote PE nodes need to use to send SRv6 encapsulated traffic towards the endpoint. The local SID directly translates into which local IP table the PE node should use to lookup for the endpoint when receiving traffic for it.

It should be noted that the endpoint does not need to attach to the PE node directly (i.e. it can attach to another network device downlink) as long as the PE node is notified (e.g. via off-band orchestration) of the following:

- o Which endpoint has been attached (i.e. Endpoint EID)
- o Which tenant network (if any) the endpoint belongs to (i.e. Endpoint IID)

For this version of the specification, it is assumed that the endpoint only attaches to a single PE node and that all traffic steering will be handled by SRv6. Therefore, for this version of the specification, a PE only register one SID per endpoint (or group of endpoints) into the Mapping System. Future versions of this specification will describe how to an endpoint can be served by more than one PE node and/or by more than one SID per PE node.

For SRv6 operation, endpoints need to be associated with a particular tag to be used for traffic steering policies. This means that the traffic addressed towards the endpoint may need to be steered through a particular path in the SRv6 domain. This draft assumes that a PE node knows the tag associated to endpoints attached to the local networks it is serving. How a PE node knows which tag corresponds to a particular endpoint is out of the scope of this document. In addition to the endpoint tag, to compute the path through the SRv6 network the loopback address of the egress PE node is also used.

Therefore, to make all this information available to the LISP-SRv6 deployment, the PE node will register into the Mapping System the mapping of endpoint address and IID to endpoint's tag, PE node local SID and PE node loopback. To do so, the egress PE node will send a Map-Register as specified in [I-D.ietf-lisp-rfc6833bis] with the appropriate IID and endpoint address as EID. The endpoint's tag, the PE node local SID and the PE node loopback are encoded using the SR LCAF described in Section 4 as an RLOC in the Map-Register.

3.3. Endpoint Resolution

When a PE node needs to encapsulate traffic from a local endpoint towards a remote endpoint served by a remote PE node, it looks up in its map-cache to find the appropriate destination SID (and SR policy) to use to reach the remote endpoint. This lookup takes into consideration the local IP table serving the local endpoint (i.e the IID of the mapping). If no entry is found for that remote endpoint, the PE node has to retrieve it from the Mapping System. To do so, it follows the procedures described in [I-D.ietf-lisp-rfc6833bis] and sends a Map-Request towards the Mapping System. In the Map-Request the PE node encodes its loopback address as ITR RLOC and as destination EID the address of the remote endpoint for which a destination SID is needed. It uses the IID associated with the local IP table serving the local endpoint that triggered the request.

What the Mapping System replies via a Map-Reply depends on how the SR policy is resolved. The Mapping System can reply with either the destination SID only (along with egress PE loopback address and endpoint tag) or with the complete SID list to be applied in the SRv6 underlay. This two options are discussed in detail in Section 3.4. Note that the Map-Reply may return a prefix as returned EID, which means all the endpoints within the prefix can be reached through the same SID. Optionally, the PE node can request to also be subscribed to updates in the endpoint(s) mapping following [I-D.ietf-lisp-pubsub].

Alternatively, it is also possible for a PE node to subscribe in advance for endpoint(s) mappings for a set (or sets) of endpoint EIDs. That way the map-cache will be pre-populated for those destination endpoint(s), avoiding the need for an on-demand Map-Request/Map-Reply. This optimization is at the cost of keeping more state in the PE node and Mapping System.

3.4. SR Policy Resolution

Besides retrieving the destination SID for a remote endpoint, a PE node also needs to find a suitable SR policy to send the traffic towards the destination SID through the SRv6 underlay. The

architecture discussed in this document offers different options to make the SR policy available at the PE node.

3.4.1. Decentralized

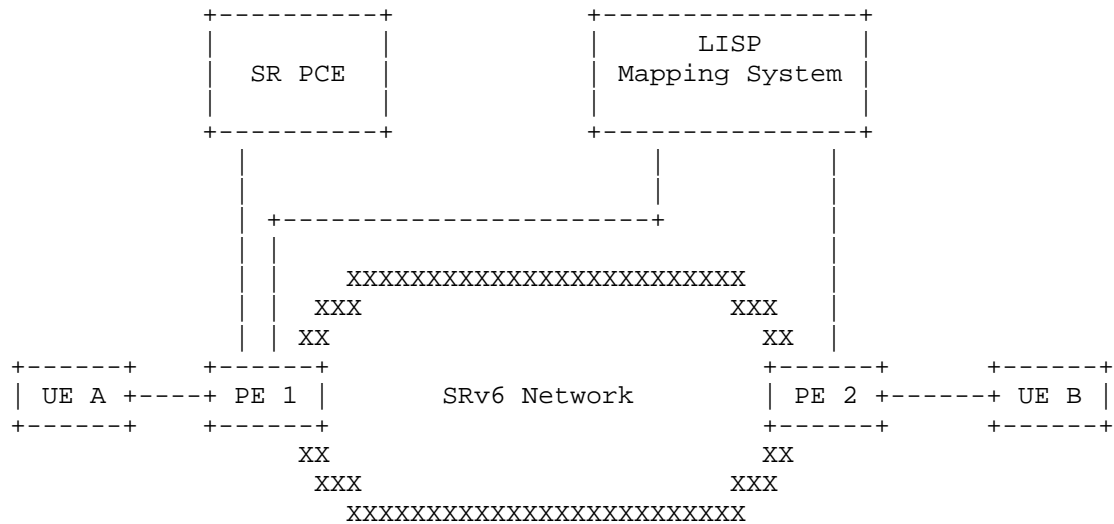


Figure 2: Decentralized SR Policy Resolution

With the decentralized approach, the SR policies are resolved independently of the endpoint resolution. In this case, the Mapping System will reply to the ingress PE node that sent the Map-Request with a Map-Reply carrying the SR LCAF described in Section 4 as RLOC. This SR LCAF contains the destination SID, egress PE loopback address, and endpoint's tag for the requested endpoint. The ingress PE will then use the loopback of the egress PE along with the tag associated with the endpoint to request a path to the SR PCE component via [RFC5440]. The SR PCE will return an SR policy (i.e. a SID list) to the ingress PE node for that egress PE node and endpoint's traffic steering tag. The ingress PE node will use that SID list received from the SR PCE (along with the destination SID retrieved from the LISP Mapping System) when encapsulating SRv6 traffic towards the endpoint.

3.4.2. Centralized

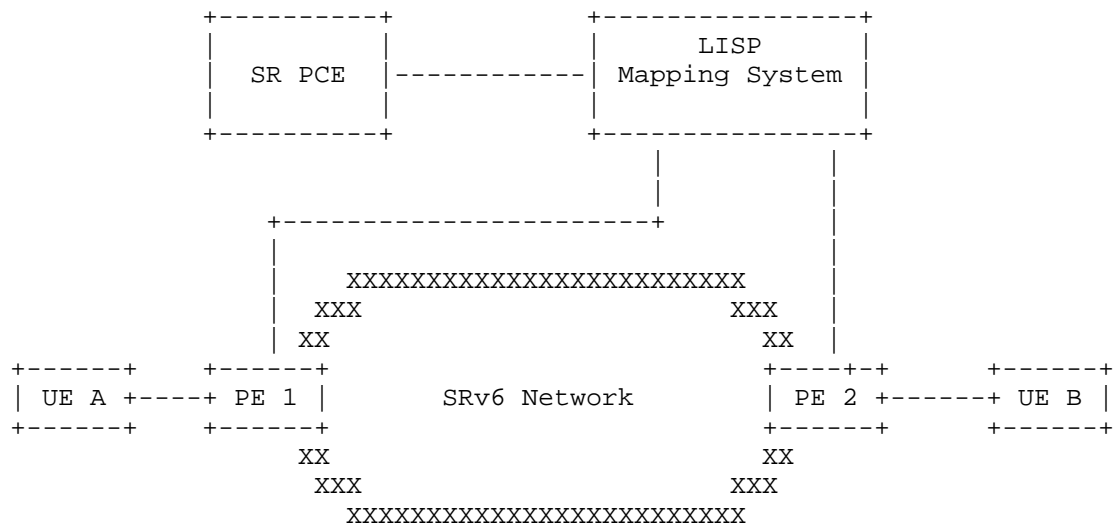


Figure 3: Centralized SR Policy Resolution

With the centralized approach the SR policies are resolved at the Mapping System when the mapping is requested by the PE node. Upon receiving a Map-Request for an endpoint from a PE node, the Mapping System queries the SR PCE to find the SR policy using [RFC5440]. To query the SR PCE, the Mapping System uses the ingress and egress PE nodes loopback addresses and the traffic steering tag of the requested endpoint. The Mapping System obtains the loopback address of the ingress PE node from the ITR-RLLOC field of the Map-Request. In this case, the Mapping System returns not only the destination SID of the remote endpoint, but also the rest of the SIDs that the traffic has to go through from the ingress PE node to the egress PE node. The SR policy SID list along with the destination egress PE node decapsulation SID are encoded as an Explicit Locator Path (ELP) [RFC8060] in the Map-Reply returned. The ingress PE node can directly use this ELP to build the SRv6 packets and does not need to query the PCE to obtain the SR policy.

3.5. Traffic Encapsulation

Once the destination SID and SR policy for a given endpoint EID are known by a PE node, the PE node will use this information to build the Segment Routing Header (SRH), if needed, and encapsulate the traffic through the SRv6 domain towards the egress PE node. Note that if there is no SR policy for a particular endpoint, no SRH is needed and the packets can be encapsulated using a vanilla IPv6 header with the destination SID as destination address. This follows

common SRv6 operation as specified in [I-D.filsfils-spring-srv6-network-programming].

When the SRv6 traffic reaches the destination, the destination SID points to the local IP table where the decapsulated traffic has to be delivered. Once decapsulated, the traffic will be routed towards the intended endpoint via lookup in that local IP table.

3.6. Endpoint Mobility

LISP-SRv6 deployment relies on the LISP mechanisms defined in [I-D.ietf-lisp-eid-mobility] to support mobility of endpoints. Following that specification, when a PE node registers an endpoint mapping, the previous PE node that had registered a mapping for the same endpoint will be notified. This serves to (1) notify the former egress PE node for the endpoint that the endpoint has moved and (2) to let former PE node know the new egress PE node for the endpoint. Once the former PE node receives the notification, it (1) stops registrations for the endpoint, (2) re-encapsulates any traffic received for the endpoint towards the new egress PE node, and (3) sends a Solicit-Map-Request message to any ingress PE node from which it receives traffic for the endpoint to let them know that they should trigger a Map-Request to update their map-caches.

In addition, if the Mapping System supports the Publish/Subscribe mechanisms described in [I-D.ietf-lisp-pubsub], a PE node can ask the Mapping System to be notified of changes in the mapping for a particular destination endpoint when it requests the mapping. This way a PE node subscribed to a particular endpoint will receive a mapping update with the new destination SID for the endpoint whenever the endpoint moves to a new PE node.

4. Segment Routing LCAF (SR LCAF)

The following Segment Routing LISP Canonical Address Format (SR LCAF) [RFC8060] is introduced to support the operation of LISP-SR deployments.

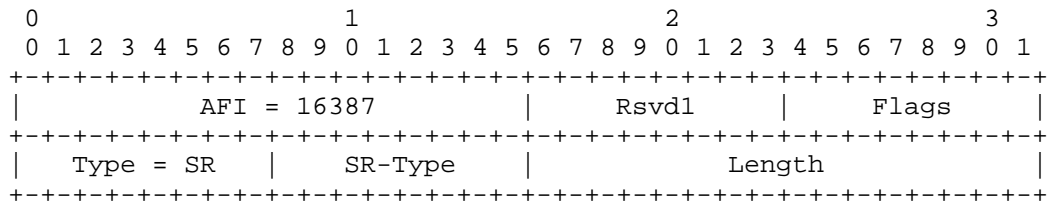


Figure 4: Segment Routing LCAF

The SR LCAF defines the SR-Type field to indicate the type of Segment Routing and the specific format of the SR LCAF. The following values are defined:

0: Reserved

1: SR-MPLS

2: SRv6

For definitions of the rest of the LCAF fields please refer to [RFC8060].

4.1. SR-MPLS

When the SR-Type is SR-MPLS (SR-Type = 1) the SR LCAF has the following format:

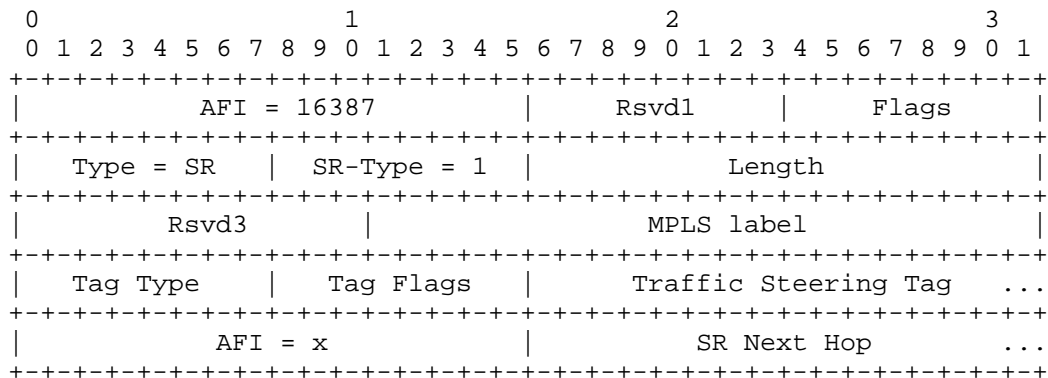


Figure 5: SR-MPLS Segment Routing LCAF

The SR-MPLS SR LCAF defines the following fields:

MPLS label: MPLS VPN label associated with the EID.

Tag Type: Type of traffic steering tag associated with the EID. Details on this traffic steering tag and different Tag Types are discussed in Section 4.3.

Tag Flags: Flags for the particular type of traffic steering tag associated with the EID.

Traffic Steering Tag: Tag associated with the EID that is used to compute SR paths.

SR Next Hop: Loopback address of the egress PE node associated with the EID.

4.2. SRv6

When the SR-Type is SR-SRv6 (SR-Type = 2) the SR LCAF has the following format:

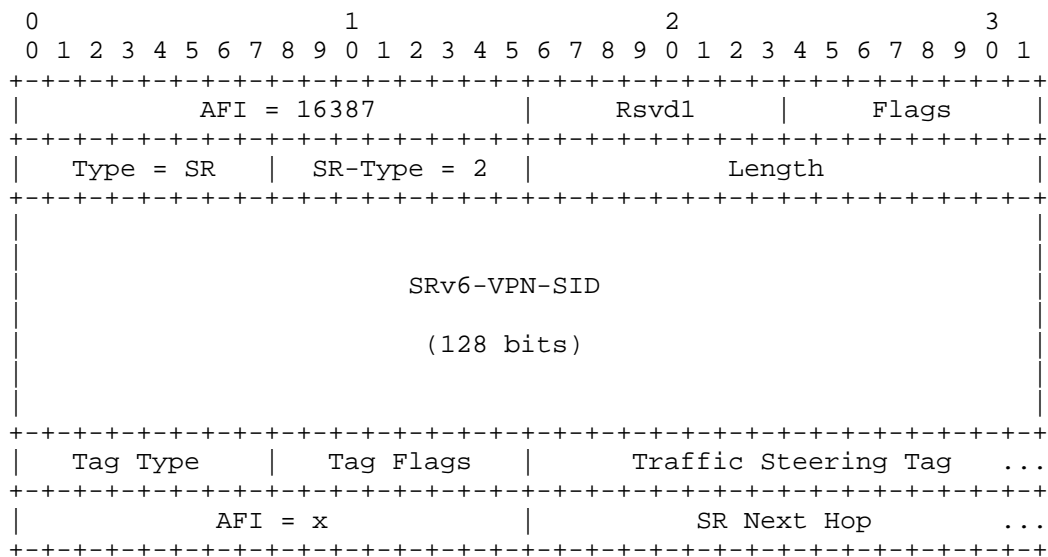


Figure 6: SRv6 Segment Routing LCAF

The SRv6 SR LCAF defines the following fields:

SRv6-VPN-SID: The SRv6 VPN SID associated with the EID.

See Section 4.1 for definitions of the rest of the fields of the SRv6 SR LCAF.

4.3. Traffic Steering Tag

The SR LCAF supports different traffic steering tags to be associated with the EID and be used in the operation of the SR deployment. The following values are defined for the Tag Type field:

0: No tag. When the Tag Type is 0 the Tag Flags are 0 and the Tag field has length of 0 octets.

1: Color. When the Tag Type is 0 the Tag Flags and Tag field have the following format.

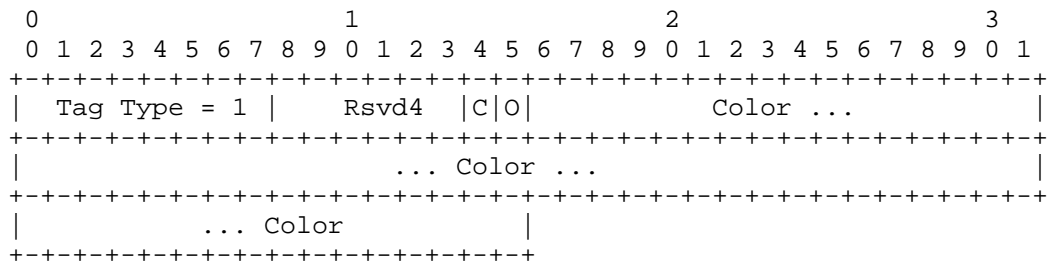


Figure 7: Color Tag

2: Slice. The tag format and flags for this Tag Type are to be defined in future versions of this document.

5. References

5.1. Normative References

- [RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <<https://www.rfc-editor.org/info/rfc4655>>.
- [RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <<https://www.rfc-editor.org/info/rfc5440>>.

[RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.

5.2. Informative References

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J.,
daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6
Network Programming", draft-filsfils-spring-srv6-network-
programming-05 (work in progress), July 2018.

[I-D.ietf-lisp-eid-mobility]
Portoles-Comeras, M., Ashtaputre, V., Moreno, V., Maino,
F., and D. Farinacci, "LISP L2/L3 EID Mobility Using a
Unified Control Plane", draft-ietf-lisp-eid-mobility-02
(work in progress), May 2018.

[I-D.ietf-lisp-pubsub]
Rodriguez-Natal, A., Ermagan, V., Leong, J., Maino, F.,
Cabellos-Aparicio, A., Barkai, S., Farinacci, D.,
Boucadair, M., Jacquenet, C., and S. Secci, "Publish/
Subscribe Functionality for LISP", draft-ietf-lisp-
pubsub-00 (work in progress), April 2018.

[I-D.ietf-lisp-rfc6833bis]
Fuller, V., Farinacci, D., and A. Cabellos-Aparicio,
"Locator/ID Separation Protocol (LISP) Control-Plane",
draft-ietf-lisp-rfc6833bis-10 (work in progress), March
2018.

Authors' Addresses

Alberto Rodriguez-Natal
Cisco Systems, Inc.
United States of America

Email: natal@cisco.com

Vina Ermagan
Cisco Systems, Inc.
United States of America

Email: vermagan@cisco.com

Fabio Maino
Cisco Systems, Inc.
United States of America

Email: fmaino@cisco.com

Darren Dukes
Cisco Systems, Inc.
Canada

Email: ddukes@cisco.com

Pablo Camarillo
Cisco Systems, Inc.
Spain

Email: pcamaril@cisco.com

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: January 27, 2021

A. Rodriguez-Natal
Cisco Systems, Inc.
V. Ermagan
Google
F. Maino
D. Dukes
P. Camarillo
C. Filsfils
Cisco Systems, Inc.
July 26, 2020

LISP Control Plane for SRv6 Endpoint Mobility
draft-rodrigueznatal-lisp-srv6-04

Abstract

This document describes the use of the LISP Control Plane to support endpoint mobility and Location/ID separation in Segment Routing v6 (SRv6) deployments.

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 January 27, 2021.

Copyright Notice

Copyright (c) 2020 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. Overview	3
3. Operation	4
3.1. Provisioning	4
3.2. Endpoint Registration	5
3.3. Endpoint Resolution	6
3.4. SR Policy Resolution	6
3.4.1. Decentralized	7
3.4.2. Centralized	7
3.5. Traffic Encapsulation	8
3.6. Endpoint Mobility	9
4. Segment Routing LCAF (SR LCAF)	9
4.1. SR-MPLS	10
4.2. SRv6	11
4.3. Traffic Steering Tag	12
5. References	12
5.1. Normative References	12
5.2. Informative References	13
Authors' Addresses	13

1. Introduction

This document defines how to use the LISP Control Plane [I-D.ietf-lisp-rfc6833bis] to support endpoint mobility and Location/ID separation in those IPv6 deployments that are using SRv6 Network Programming [I-D.ietf-spring-srv6-network-programming]. The LISP control plane is used to lookup "where" an endpoint is located, while SRv6 specifies "how" to program the SRv6 network infrastructure to transport traffic to that location.

To enable this, the egress Provider Edge (PE) nodes register via LISP control plane their local SRv6 Segment IDs (SIDs) to be used to reach endpoints attached to them. Ingress PE nodes retrieve via LISP control plane this mapping information, and use SRv6 network programming to encapsulate and steer the traffic towards the destination egress PE node.

The LISP control plane provides on-demand endpoint-to-SID mapping, as well as support for anchorless dynamic endpoint mobility.

2. Overview

The ingress PE nodes receive traffic from endpoints in their local networks, encapsulate it in IPv6 packets following SRv6 policies and forward it to the SRv6 domain. Similarly, egress PE nodes receive SRv6 traffic from the SRv6 domain, remove the SRv6 encapsulation and forward the traffic to one of their locally attached networks according to the decapsulation SID received in the packets. Ingress PE nodes and egress PE nodes can be co-located in the same node, in this document we use "PE node" to refer to those collocated nodes.

This specification leverages the LISP Mapping System to store mappings of endpoints to decapsulation SIDs. Endpoints are neither LISP nor SRv6 aware and can roam freely across different PE nodes. The mappings in the LISP Mapping System can be used by PE nodes to know to which PE node an endpoint is currently connected. The LISP Mapping System is composed of LISP Map-Resolvers and Map-Servers but, for convenience, this document refers to the Mapping System as a single logical entity.

To use the LISP Mapping System, in this specification the PE nodes also implement the control-plane logic of LISP Ingress/Egress Tunnel Routers (xTRs). As a LISP xTR, a PE node keeps a local database of all the endpoints that are attached to its local network(s). It also keeps a list of local decapsulation function SIDs that map to each of its local network(s). A PE node registers into the LISP Mapping System its local endpoint-to-SID mappings. These mappings also contain the traffic steering tag associated with the endpoint. In addition to its local mappings, a PE node also keeps a local map-cache of remote endpoint-to-SID mappings that it uses to find the SID to use to encapsulate traffic towards a remote endpoint.

This specification also assumes an SR Path Computation Element (PCE) [RFC4655] that can compute and provide SRv6 paths from a given ingress PE node to a given egress PE node. The paths are based on the ingress and egress PE nodes and on the traffic steering tag that is associated with the destination endpoint. Figure 1 shows an example diagram to be used as a reference for the architecture described in this document.

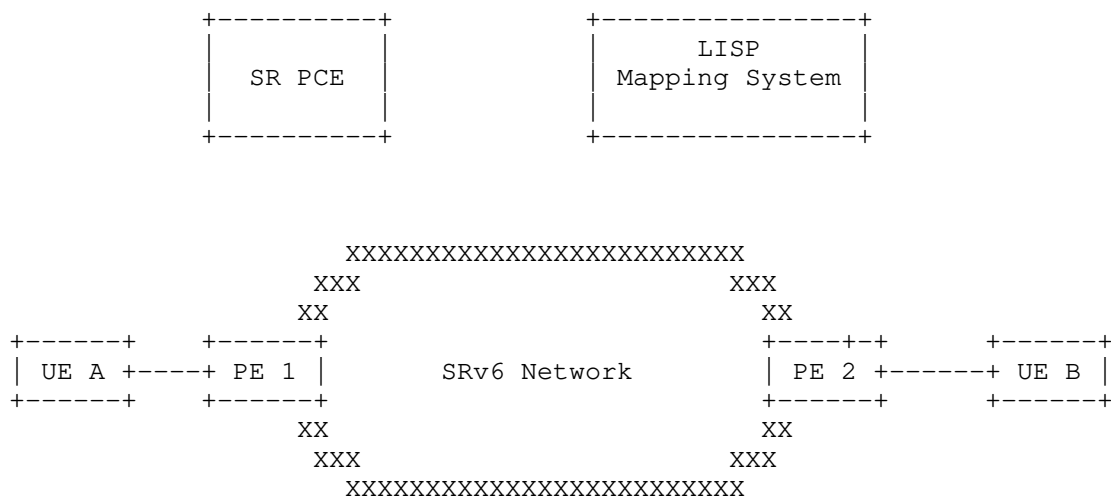


Figure 1: Reference Architecture

3. Operation

3.1. Provisioning

Each PE node is connected to the SRv6 domain and to one or more local networks. These local networks may represent different tenants/VPNs. Each tenant network is globally identified via a unique Instance Identifier (IID). Locally, these tenant networks are identified at each PE node by the local IP table assigned to them. While the IID for a tenant network is global across the domain, the local IP table assigned to it in a given PE node is local to that PE node. Each PE node needs to be provisioned with the one-to-one mapping of global IID to local IP table per each tenant network it is serving. The provisioning of the IID to the local IP table is out of the scope of this document. PE nodes use this IID to local IP table information to know which IID they need to use when requesting mappings for traffic coming from a particular tenant network (i.e. from a particular local IP table).

Per each local IP table, each PE node has a local SRv6 "End.DT46" function (see [I-D.ietf-spring-srv6-network-programming]) that decapsulates SRv6 traffic and forwards the inner traffic for lookup into that particular IP table. The PE node concatenates one of its local SRv6 locators with each of these decapsulation functions to create SIDs that point to each of the different tenant networks it is serving. These SIDs are kept onto the "My local SIDs table" of the

PE node and they are used to decapsulate incoming SRv6 traffic into the appropriate local IP table.

Beyond SRv6 state, each PE node has to be provisioned with the address of at least one Map-Resolver it will use to retrieve remote endpoint-to-SID mappings from the Mapping System. Similarly, it has to be provisioned with the address of the Map-Server to which it is going to register their local endpoint-to-SID mappings.

3.2. Endpoint Registration

Endpoints attach to the local networks served by PE nodes. Upon attachment of an endpoint, the PE node will correlate the local IP table that corresponds to the local network where the endpoint attached to a global IID. It does so by using its local information of global IID to local IP table. The local IP table also correlates with the local SID that remote PE nodes need to use to send SRv6 encapsulated traffic towards the endpoint. The local SID directly translates into which local IP table the PE node should use to lookup for the endpoint when receiving traffic for it.

It should be noted that the endpoint does not need to attach to the PE node directly (i.e. it can attach to another network device downlink) as long as the PE node is notified (e.g. via off-band orchestration) of the following:

- o Which endpoint has been attached (i.e. Endpoint EID)
- o Which tenant network (if any) the endpoint belongs to (i.e. Endpoint IID)

For this version of the specification, it is assumed that the endpoint only attaches to a single PE node and that all traffic steering will be handled by SRv6. Therefore, for this version of the specification, a PE only register one SID per endpoint (or group of endpoints) into the Mapping System. Future versions of this specification will describe how to an endpoint can be served by more than one PE node and/or by more than one SID per PE node.

For SRv6 operation, endpoints need to be associated with a particular tag to be used for traffic steering policies. This means that the traffic addressed towards the endpoint may need to be steered through a particular path in the SRv6 domain. This draft assumes that a PE node knows the tag associated to endpoints attached to the local networks it is serving. How a PE node knows which tag corresponds to a particular endpoint is out of the scope of this document. In addition to the endpoint tag, to compute the path through the SRv6 network the loopback address of the egress PE node is also used.

Therefore, to make all this information available to the LISP-SRv6 deployment, the PE node will register into the Mapping System the mapping of endpoint address and IID to endpoint's tag, PE node local SID and PE node loopback. To do so, the egress PE node will send a Map-Register as specified in [I-D.ietf-lisp-rfc6833bis] with the appropriate IID and endpoint address as EID. The endpoint's tag, the PE node local SID and the PE node loopback are encoded using the SR LCAF described in Section 4 as an RLOC in the Map-Register.

3.3. Endpoint Resolution

When a PE node needs to encapsulate traffic from a local endpoint towards a remote endpoint served by a remote PE node, it looks up in its map-cache to find the appropriate destination SID (and SR policy) to use to reach the remote endpoint. This lookup takes into consideration the local IP table serving the local endpoint (i.e the IID of the mapping). If no entry is found for that remote endpoint, the PE node has to retrieve it from the Mapping System. To do so, it follows the procedures described in [I-D.ietf-lisp-rfc6833bis] and sends a Map-Request towards the Mapping System. In the Map-Request the PE node encodes its loopback address as ITR RLOC and as destination EID the address of the remote endpoint for which a destination SID is needed. It uses the IID associated with the local IP table serving the local endpoint that triggered the request.

What the Mapping System replies via a Map-Reply depends on how the SR policy is resolved. The Mapping System can reply with either the destination SID only (along with egress PE loopback address and endpoint tag) or with the complete SID list to be applied in the SRv6 underlay. This two options are discussed in detail in Section 3.4. Note that the Map-Reply may return a prefix as returned EID, which means all the endpoints within the prefix can be reached through the same SID. Optionally, the PE node can request to also be subscribed to updates in the endpoint(s) mapping following [I-D.ietf-lisp-pubsub].

Alternatively, it is also possible for a PE node to subscribe in advance for endpoint(s) mappings for a set (or sets) of endpoint EIDs. That way the map-cache will be pre-populated for those destination endpoint(s), avoiding the need for an on-demand Map-Request/Map-Reply. This optimization is at the cost of keeping more state in the PE node and Mapping System.

3.4. SR Policy Resolution

Besides retrieving the destination SID for a remote endpoint, a PE node also needs to find a suitable SR policy to send the traffic towards the destination SID through the SRv6 underlay. The

architecture discussed in this document offers different options to make the SR policy available at the PE node.

3.4.1. Decentralized

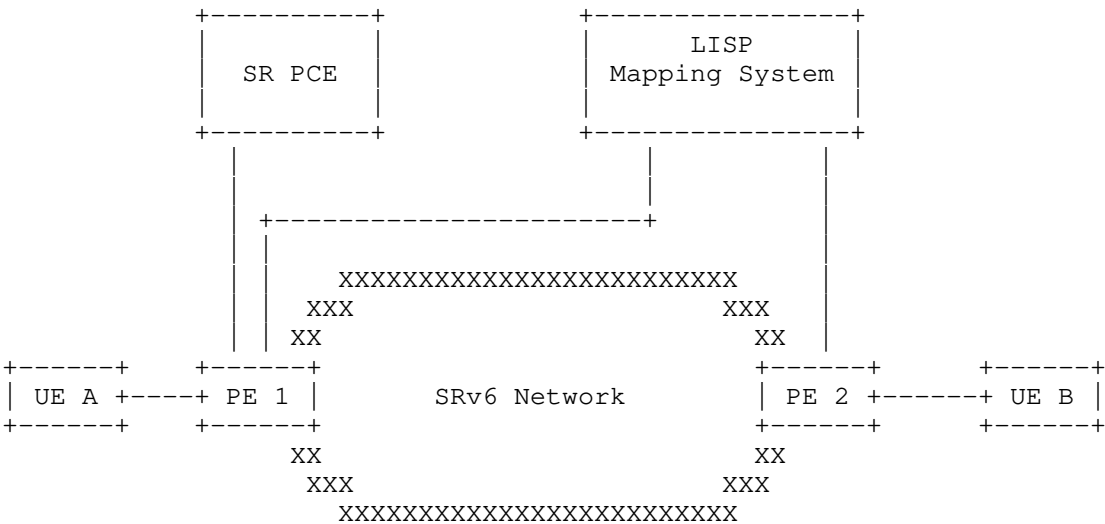


Figure 2: Decentralized SR Policy Resolution

With the decentralized approach, the SR policies are resolved independently of the endpoint resolution. In this case, the Mapping System will reply to the ingress PE node that sent the Map-Request with a Map-Reply carrying the SR LCAF described in Section 4 as RLOC. This SR LCAF contains the destination SID, egress PE loopback address, and endpoint's tag for the requested endpoint. The ingress PE will then use the loopback of the egress PE along with the tag associated with the endpoint to request a path to the SR PCE component via [RFC5440]. The SR PCE will return an SR policy (i.e. a SID list) to the ingress PE node for that egress PE node and endpoint's traffic steering tag. The ingress PE node will use that SID list received from the SR PCE (along with the destination SID retrieved from the LISP Mapping System) when encapsulating SRv6 traffic towards the endpoint.

3.4.2. Centralized

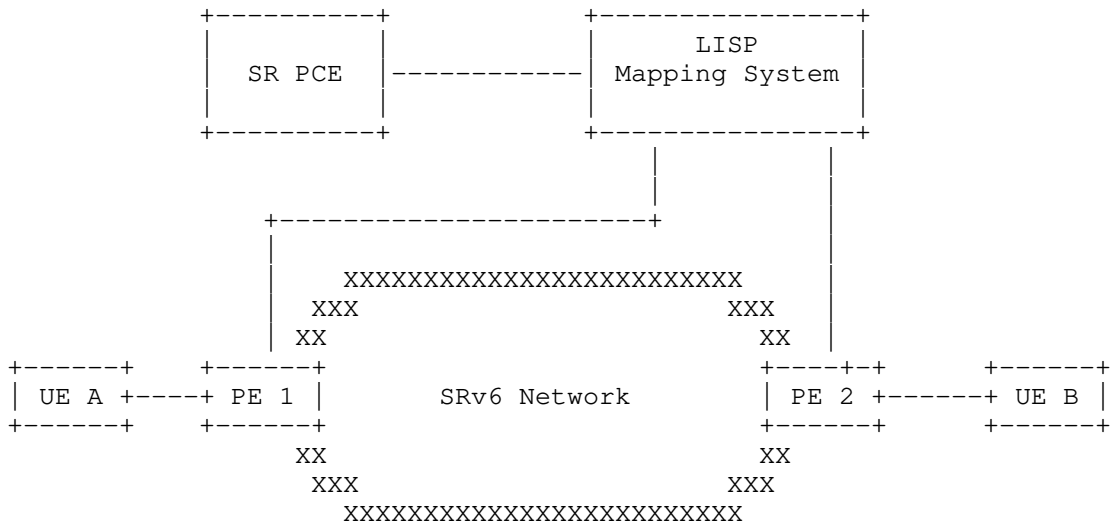


Figure 3: Centralized SR Policy Resolution

With the centralized approach the SR policies are resolved at the Mapping System when the mapping is requested by the PE node. Upon receiving a Map-Request for an endpoint from a PE node, the Mapping System queries the SR PCE to find the SR policy using [RFC5440]. To query the SR PCE, the Mapping System uses the ingress and egress PE nodes loopback addresses and the traffic steering tag of the requested endpoint. The Mapping System obtains the loopback address of the ingress PE node from the ITR-RLOC field of the Map-Request. In this case, the Mapping System returns not only the destination SID of the remote endpoint, but also the rest of the SIDs that the traffic has to go through from the ingress PE node to the egress PE node. The SR policy SID list along with the destination egress PE node decapsulation SID are encoded as an Explicit Locator Path (ELP) [RFC8060] in the Map-Reply returned. The ingress PE node can directly use this ELP to build the SRv6 packets and does not need to query the PCE to obtain the SR policy.

3.5. Traffic Encapsulation

Once the destination SID and SR policy for a given endpoint EID are known by a PE node, the PE node will use this information to build the Segment Routing Header (SRH), if needed, and encapsulate the traffic through the SRv6 domain towards the egress PE node. Note that if there is no SR policy for a particular endpoint, no SRH is needed and the packets can be encapsulated using a vanilla IPv6 header with the destination SID as destination address. This follows

common SRv6 operation as specified in [I-D.ietf-spring-srv6-network-programming].

When the SRv6 traffic reaches the destination, the destination SID points to the local IP table where the decapsulated traffic has to be delivered. Once decapsulated, the traffic will be routed towards the intended endpoint via lookup in that local IP table.

3.6. Endpoint Mobility

LISP-SRv6 deployment relies on the LISP mechanisms defined in [I-D.ietf-lisp-eid-mobility] to support mobility of endpoints. Following that specification, when a PE node registers an endpoint mapping, the previous PE node that had registered a mapping for the same endpoint will be notified. This serves to (1) notify the former egress PE node for the endpoint that the endpoint has moved and (2) to let former PE node know the new egress PE node for the endpoint. Once the former PE node receives the notification, it (1) stops registrations for the endpoint, (2) re-encapsulates any traffic received for the endpoint towards the new egress PE node, and (3) sends a Solicit-Map-Request message to any ingress PE node from which it receives traffic for the endpoint to let them know that they should trigger a Map-Request to update their map-caches.

In addition, if the Mapping System supports the Publish/Subscribe mechanisms described in [I-D.ietf-lisp-pubsub], a PE node can ask the Mapping System to be notified of changes in the mapping for a particular destination endpoint when it requests the mapping. This way a PE node subscribed to a particular endpoint will receive a mapping update with the new destination SID for the endpoint whenever the endpoint moves to a new PE node.

4. Segment Routing LCAF (SR LCAF)

The following Segment Routing LISP Canonical Address Format (SR LCAF) [RFC8060] is introduced to support the operation of LISP-SR deployments.

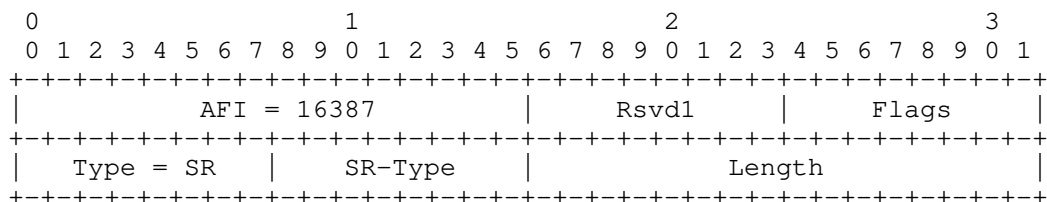


Figure 4: Segment Routing LCAF

The SR LCAF defines the SR-Type field to indicate the type of Segment Routing and the specific format of the SR LCAF. The following values are defined:

0: Reserved

1: SR-MPLS

2: SRv6

For definitions of the rest of the LCAF fields please refer to [RFC8060].

4.1. SR-MPLS

When the SR-Type is SR-MPLS (SR-Type = 1) the SR LCAF has the following format:

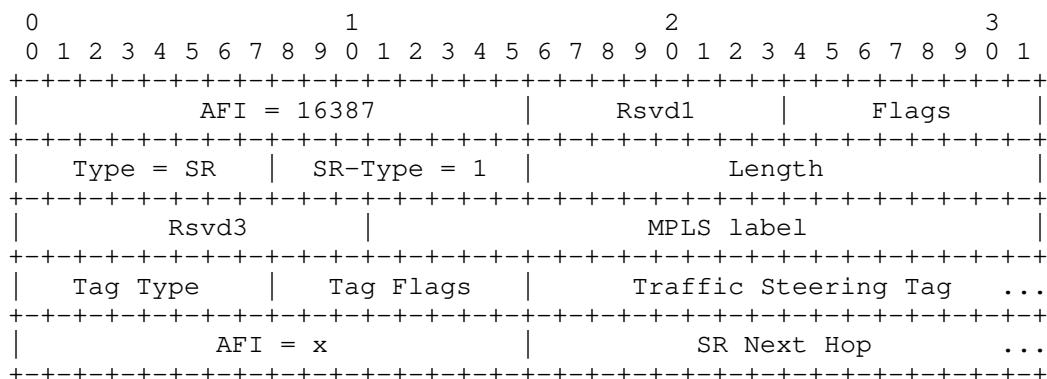


Figure 5: SR-MPLS Segment Routing LCAF

The SR-MPLS SR LCAF defines the following fields:

MPLS label: MPLS VPN label associated with the EID.

Tag Type: Type of traffic steering tag associated with the EID. Details on this traffic steering tag and different Tag Types are discussed in Section 4.3.

Tag Flags: Flags for the particular type of traffic steering tag associated with the EID.

Traffic Steering Tag: Tag associated with the EID that is used to compute SR paths.

SR Next Hop: Loopback address of the egress PE node associated with the EID.

4.2. SRv6

When the SR-Type is SR-SRv6 (SR-Type = 2) the SR LCAF has the following format:

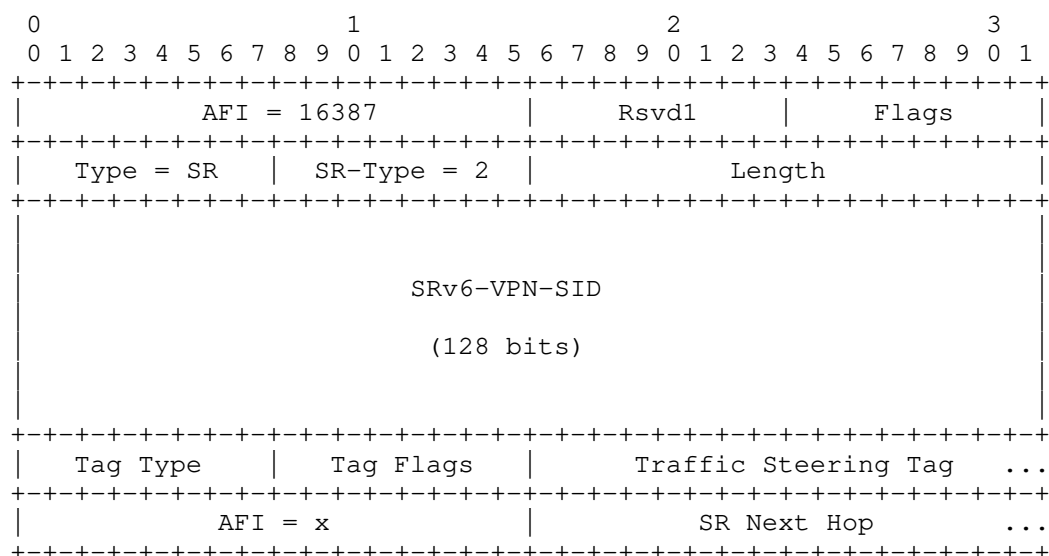


Figure 6: SRv6 Segment Routing LCAF

The SRv6 SR LCAF defines the following fields:

SRv6-VPN-SID: The SRv6 VPN SID associated with the EID.

See Section 4.1 for definitions of the rest of the fields of the SRv6 SR LCAF.

4.3. Traffic Steering Tag

The SR LCAF supports different traffic steering tags to be associated with the EID and be used in the operation of the SR deployment. The following values are defined for the Tag Type field:

0: No tag. When the Tag Type is 0 the Tag Flags are 0 and the Tag field has length of 0 octets.

1: Color. When the Tag Type is 0 the Tag Flags and Tag field have the following format.

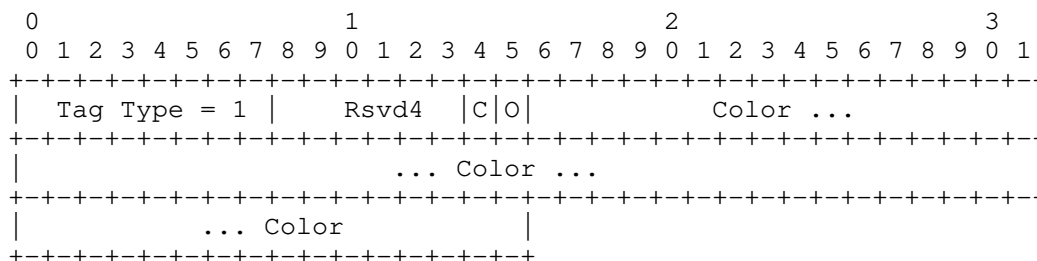


Figure 7: Color Tag

2: Slice. The tag format and flags for this Tag Type are to be defined in future versions of this document.

5. References

5.1. Normative References

- [RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <<https://www.rfc-editor.org/info/rfc4655>>.
- [RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <<https://www.rfc-editor.org/info/rfc5440>>.

[RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <<https://www.rfc-editor.org/info/rfc8060>>.

5.2. Informative References

[I-D.ietf-lisp-eid-mobility]
Portoles-Comeras, M., Ashtaputre, V., Moreno, V., Maino, F., and D. Farinacci, "LISP L2/L3 EID Mobility Using a Unified Control Plane", draft-ietf-lisp-eid-mobility-06 (work in progress), May 2020.

[I-D.ietf-lisp-pubsub]
Rodriguez-Natal, A., Ermagan, V., Cabellos-Aparicio, A., Barkai, S., and M. Boucadair, "Publish/Subscribe Functionality for LISP", draft-ietf-lisp-pubsub-06 (work in progress), July 2020.

[I-D.ietf-lisp-rfc6833bis]
Farinacci, D., Maino, F., Fuller, V., and A. Cabellos-Aparicio, "Locator/ID Separation Protocol (LISP) Control-Plane", draft-ietf-lisp-rfc6833bis-27 (work in progress), January 2020.

[I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-16 (work in progress), June 2020.

Authors' Addresses

Alberto Rodriguez-Natal
Cisco Systems, Inc.
United States of America

Email: natal@cisco.com

Vina Ermagan
Google
United States of America

Email: ermagan@gmail.com

Fabio Maino
Cisco Systems, Inc.
United States of America

Email: fmaino@cisco.com

Darren Dukes
Cisco Systems, Inc.
Canada

Email: ddukes@cisco.com

Pablo Camarillo
Cisco Systems, Inc.
Spain

Email: pcamaril@cisco.com

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 2, 2018

F. Templin, Ed.
G. Saccone
Boeing Research & Technology
G. Dawra
LinkedIn
A. Lindem
Cisco Systems, Inc.
May 31, 2018

A Simple BGP-based Mobile Routing System for the Aeronautical
Telecommunications Network
draft-templin-atn-bgp-07.txt

Abstract

The International Civil Aviation Organization (ICAO) is investigating mobile routing solutions for a worldwide Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS). The ATN/IPS will eventually replace existing communication services with an IPv6-based service supporting pervasive Air Traffic Management (ATM) for Air Traffic Controllers (ATC), Airline Operations Controllers (AOC), and all commercial aircraft worldwide. This informational document describes a simple and extensible mobile routing service based on industry-standard BGP to address the ATN/IPS requirements.

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 December 2, 2018.

Copyright Notice

Copyright (c) 2018 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. Terminology	5
3. ATN/IPS Routing System	6
4. ATN/IPS Radio Access Network (RAN) Model	9
5. ATN/IPS Route Optimization	11
6. BGP Protocol Considerations	13
7. Implementation Status	14
8. IANA Considerations	14
9. Security Considerations	14
10. Acknowledgements	15
11. References	15
11.1. Normative References	15
11.2. Informative References	16
Authors' Addresses	17

1. Introduction

The worldwide Air Traffic Management (ATM) system today uses a service known as Aeronautical Telecommunications Network based on Open Systems Interconnection (ATN/OSI). The service is used to augment controller to pilot voice communications with rudimentary short text command and control messages. The service has seen successful deployment in a limited set of worldwide ATM domains.

The International Civil Aviation Organization [ICAO] is now undertaking the development of a next-generation replacement for ATN/OSI known as Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS). ATN/IPS will eventually provide an IPv6-based [RFC8200] service supporting pervasive ATM for Air Traffic Controllers (ATC), Airline Operations Controllers (AOC), and all commercial aircraft worldwide. As part of the ATN/IPS undertaking, a

new mobile routing service will be needed. This document presents an approach based on the Border Gateway Protocol (BGP) [RFC4271].

Aircraft communicate via wireless aviation data links that typically support much lower data rates than terrestrial wireless and wired-line communications. For example, some Very High Frequency (VHF)-based data links only support data rates on the order of 32Kbps and an emerging L-Band data link that is expected to play a key role in future aeronautical communications only supports rates on the order of 1Mbps. Although satellite data links can provide much higher data rates during optimal conditions, like any other aviation data link they are subject to errors, delay, disruption, signal intermittence, degradation due to atmospheric conditions, etc. The well-connected ground domain ATN/IPS network should therefore treat each safety-of-flight critical packet produced by (or destined to) an aircraft as a precious commodity and strive for an optimized service that provides the highest possible degree of reliability.

The ATN/IPS is an IPv6-based overlay network that assumes a worldwide connected Internetworking underlay for carrying tunneled ATM communications. The Internetworking underlay could be manifested as a private collection of long-haul backbone links (e.g., fiberoptics, copper, SATCOM, etc.) interconnected by high-performance networking gear such as bridges, switches, and routers. Such a private network would need to connect all ATN/IPS participants worldwide, and could therefore present a considerable cost for a large-scale deployment of new infrastructure. Alternatively, the ATN/IPS could be deployed as a secured overlay over the existing global public Internet. For example, ATN/IPS nodes could be deployed as part of an SD-WAN or an MPLS-WAN that rides over the public Internet via secured tunnels.

The ATN/IPS further assumes that each aircraft will receive an IPv6 Mobile Network Prefix (MNP) that accompanies the aircraft wherever it travels. ATCs and AOCs will likewise receive IPv6 prefixes, but they would typically appear in static (not mobile) deployments such as air traffic control towers, airline headquarters, etc. Throughout the rest of this document, we therefore use the term "MNP" when discussing an IPv6 prefix that is delegated to any ATN/IPS end system, including ATCs, AOCs, and aircraft. We also use the term Mobility Service Prefix (MSP) to refer to an aggregated prefix assigned to the ATN/IPS by an Internet assigned numbers authority, and from which all MNPs are delegated (e.g., up to 2**32 IPv6 /64 MNPs could be delegated from the MSP 2001:db8::/32).

Connexion By Boeing [CBB] was an early aviation mobile routing service based on dynamic updates in the global public Internet BGP routing system. Practical experience with the approach has shown that frequent injections and withdrawals of MNPs in the Internet

routing system can result in excessive BGP update messaging, slow routing table convergence times, and extended outages when no route is available. This is due to both conservative default BGP protocol timing parameters (see Section 6) and the complex peering interconnections of BGP routers within the global Internet infrastructure. The situation is further exacerbated by frequent aircraft mobility events that each result in BGP updates that must be propagated to all BGP routers in the Internet that carry a full routing table.

We therefore consider an approach using a BGP overlay network routing system where a private BGP routing protocol instance is maintained between ATN/IPS Autonomous System (AS) Border Routers (ASBRs). The private BGP instance does not interact with the native BGP routing system in the connected Internetworking underlay, and BGP updates are unidirectional from "stub" ASBRs (s-ASBRs) to a small set of "core" ASBRs (c-ASBRs) in a hub-and-spokes topology. No extensions to the BGP protocol are necessary.

The s-ASBRs for each stub AS connect to a small number of c-ASBRs via dedicated high speed links and/or tunnels across the Internetworking underlay using industry-standard encapsulations (e.g., Generic Routing Encapsulation (GRE) [RFC2784], IPsec [RFC4301], etc.). In particular, tunneling must be used when neighboring ASBRs are separated by many Internetworking underlay hops.

The s-ASBRs engage in external BGP (eBGP) peerings with their respective c-ASBRs, and only maintain routing table entries for the MNPs currently active within the stub AS. The s-ASBRs send BGP updates for MNP injections or withdrawals to c-ASBRs but do not receive any BGP updates from c-ASBRs. Instead, the s-ASBRs maintain default routes with their c-ASBRs as the next hop, and therefore hold only partial topology information.

The c-ASBRs connect to other c-ASBRs using internal BGP (iBGP) peerings over which they collaboratively maintain a full routing table for all active MNPs currently in service. Therefore, only the c-ASBRs maintain a full BGP routing table and never send any BGP updates to s-ASBRs. This simple routing model therefore greatly reduces the number of BGP updates that need to be synchronized among peers, and the number is reduced further still when intradomain routing changes within stub ASes are processed within the AS instead of being propagated to the core. BGP Route Reflectors (RRs) [RFC4456] can also be used to support increased scaling properties.

The remainder of this document discusses the proposed BGP-based ATN/IPS mobile routing service.

2. Terminology

The terms Autonomous System (AS) and Autonomous System Border Router (ASBR) are the same as defined in [RFC4271].

The following terms are defined for the purposes of this document:

Air Traffic Managemnet (ATM)

The worldwide service for coordinating safe aviation operations.

Air Traffic Controller (ATC)

A government agent responsible for coordinating with aircraft within a defined operational region via voice and/or data Command and Control messaging.

Airline Operations Controller (AOC)

An airline agent responsible for tracking and coordinating with aircraft within their fleet.

Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS)

A future aviation network for ATCs and AOCs to coordinate with all aircraft operating worldwide. The ATN/IPS will be an IPv6-based overlay network service that connects access networks via tunneling over an Internetworking underlay.

Internetworking underlay A connected wide-area network that supports overlay network tunneling and connects Radio Access Networks to the rest of the ATN/IPS.

Radio Access Network (RAN)

An aviation radio data link service provider's network, including radio transmitters and receivers as well as supporting ground-domain infrastructure needed to convey a customer's data packets to the outside world. The term RAN is intended in the same spirit as for cellular operator networks and other radio-based Internet service provider networks. For simplicity, we also use the term RAN to refer to ground-domain networks that connect AOCs and ATCs without any aviation radio communications.

Core Autonomous System Border Router (c-ASBR) A BGP router located in the hub of the ATN/IPS hub-and-spokes overlay network topology.

Core Autonomous System The "hub" autonomous system maintained by all c-ASBRs.

Stub Autonomous System Border Router (s-ASBR) A BGP router configured as a spoke in the ATN/IPS hub-and-spokes overlay network topology.

Stub Autonomous System A logical grouping that includes all Clients currently associated with a given s-ASBR.

Client An ATC, AOC or aircraft that connects to the ATN/IPS as a leaf node. The Client could be a singleton host, or a router that connects a mobile network.

Proxy A node at the edge of a RAN that acts as an intermediary between Clients and s-ASBRs. From the Client's perspective, the Proxy presents the appearance that the Client is communicating directly with the s-ASBR. From the s-ASBR's perspective, the Proxy presents the appearance that the s-ASBR is communicating directly with the Client.

Mobile Network Prefix (MNP) An IPv6 prefix that is delegated to any ATN/IPS end system, including ATCs, AOCs, and aircraft.

Mobility Service Prefix (MSP) An aggregated prefix assigned to the ATN/IPS by an Internet assigned numbers authority, and from which all MNPs are delegated (e.g., up to 2^{32} IPv6 /64 MNPs could be delegated from the MSP 2001:db8::/32).

3. ATN/IPS Routing System

The proposed ATN/IPS routing system comprises a private BGP instance coordinated in an overlay network via tunnels between neighboring ASBRs over the Internetworking underlay. The overlay does not interact with the native BGP routing system in the connected underlying Internetwork, and each c-ASBR advertises only a small and unchanging set of MSPs into the Internetworking underlay routing system instead of the full dynamically changing set of MNPs. (For example, when the Internetworking underlay is the global public Internet the c-ASBRs advertise the MSPs in the public BGP Internet routing system.)

In a reference deployment, one or more s-ASBRs connect each stub AS to the overlay using a shared stub AS Number (ASN). Each s-ASBR further uses eBGP to peer with one or more c-ASBRs. All c-ASBRs are members of the same core AS, and use a shared core ASN. Since the private BGP instance is separate from the global public Internet BGP routing system, the ASBRs can use either a private ASN per [RFC6996] or simply use public ASNs noting that the ASNs may overlap with those already assigned in the Internet. (A third alternative would be to

procure globally-unique public ASNs, but cost and maintenance requirements must be considered.)

The c-ASBRs use iBGP to maintain a synchronized consistent view of all active MNPs currently in service. Figure 1 below represents the reference deployment. (Note that the figure shows details for only two s-ASBRs (s-ASBR1 and s-ASBR2) due to space constraints, but the other s-ASBRs should be understood to have similar Stub AS, MNP and eBGP peering arrangements.) The solution described in this document is flexible enough to extend to these topologies.

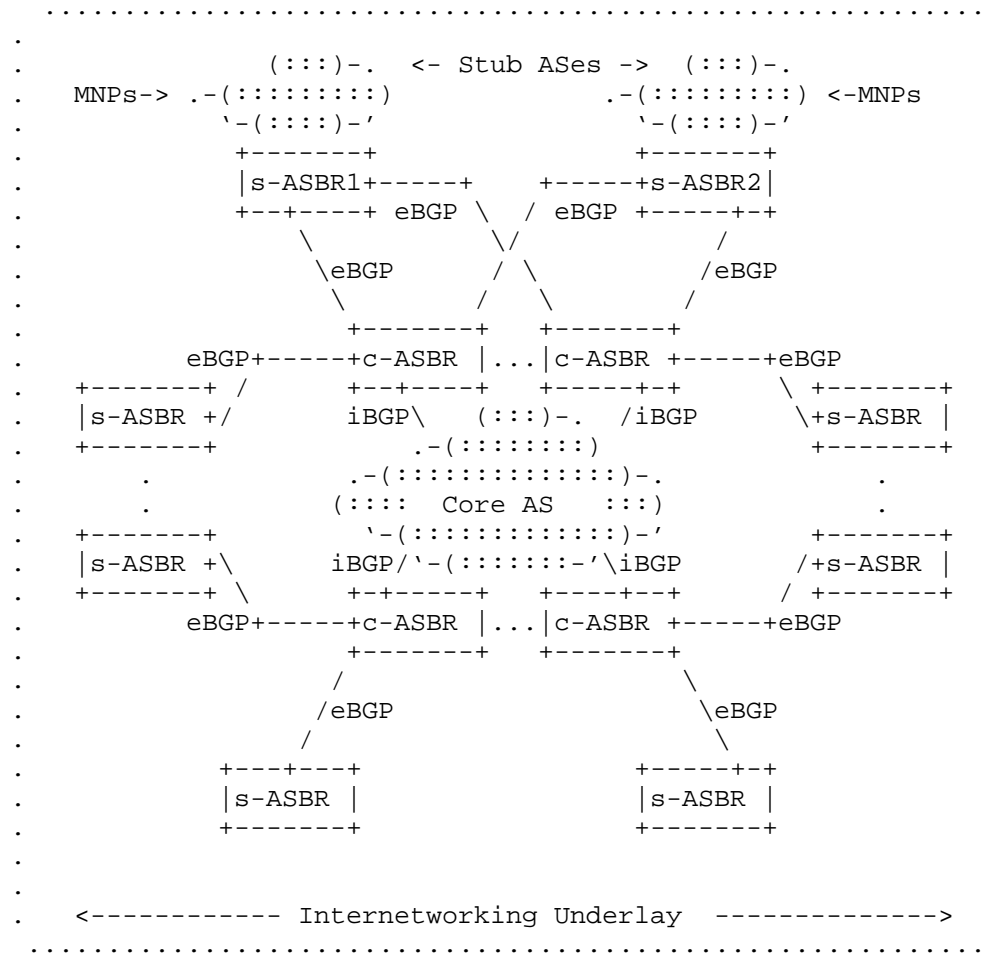


Figure 1: Reference Deployment

In the reference deployment, each s-ASBR maintains routes for active MNPs that currently belong to its stub AS. In response to "Inter-domain" mobility events, each s-ASBR will dynamically announce new MNPs and withdraws departed MNPs in its eBGP updates to c-ASBRs. Since ATN/IPS end systems are expected to remain within the same stub AS for extended timeframes, however, intra-domain mobility events (such as an aircraft handing off between cell towers) are handled within the stub AS instead of being propagated as inter-domain eBGP updates.

Each c-ASBR configures a black-hole route for each of its MSPs. By black-holing the MSPs, the c-ASBR will maintain forwarding table entries only for the MNPs that are currently active, and packets destined to all other MNPs will correctly incur ICMPv6 Destination Unreachable messages [RFC4443] due to the black hole route. (This is the same behavior as for ordinary BGP routers in the Internet when they receive packets for which there is no route available.) The c-ASBRs do not send eBGP updates for MNPs to s-ASBRs, but instead originate a default route. In this way, s-ASBRs have only partial topology knowledge (i.e., they know only about the active MNPs currently within their stub ASes) and they forward all other packets to c-ASBRs which have full topology knowledge.

Scaling properties of this ATN/IPS routing system are limited by the number of BGP routes that can be carried by the c-ASBRs. A 2015 study showed that BGP routers in the global public Internet at that time carried more than 500K routes with linear growth and no signs of router resource exhaustion [BGP]. A more recent network emulation study also showed that a single c-ASBR can accommodate at least 1M dynamically changing BGP routes even on a lightweight virtual machine. Commercially-available high-performance dedicated router hardware can support many millions of routes.

Therefore, assuming each c-ASBR can carry 1M or more routes, this means that at least 1M ATN/IPS end system MNPs can be serviced by a single set of c-ASBRs and that number could be further increased by using RRs. Another means of increasing scale would be to assign a different set of c-ASBRs for each set of MSPs. In that case, each s-ASBR still peers with one or more c-ASBRs from each set of c-ASBRs, but the s-ASBR institutes route filters so that it only sends BGP updates to the specific set of c-ASBRs that aggregate the MSP. For example, if the MSP for the ATN/IPS deployment is 2001:db8::/32, a first set of c-ASBRs could service the MSP segment 2001:db8::/40, a second set could service 2001:db8:0100::/40, a third set could service 2001:db8:0200::/40, etc.

In this way, each set of c-ASBRs services a specific set of MSPs that they inject into the Internetworking underlay native routing system,

and each s-ASBR configures MSP-specific routes that list the correct set of c-ASBRs as next hops. This BGP routing design also allows for natural incremental deployment, and can support initial small-scale deployments followed by dynamic deployment of additional ATN/IPS infrastructure elements without disturbing the already-deployed base.

4. ATN/IPS Radio Access Network (RAN) Model

Radio Access Networks (RANs) connect end system Clients such as aircraft, ATCs, AOCs etc. to the ATN/IPS routing system. Clients may connect to multiple RANs at once, for example, when they have both satellite and cellular data links activated simultaneously. Clients may further move between RANs in a manner that is perceived as a network layer mobility event. Clients could therefore employ a multilink/mobility routing service such as that discussed in [I-D.templin-aerolink].

Clients register all of their active data link connections with their serving s-ASBRs as discussed in Section 3. Clients may connect to s-ASBRs either directly, or via a Proxy at the edge of the RAN.

Figure 2 shows the ATN/IPS RAN model where Clients connect to RANs via aviation data links. Clients register their RAN addresses with a nearby s-ASBR, where the registration process may be brokered by a Proxy at the edge of the RAN.

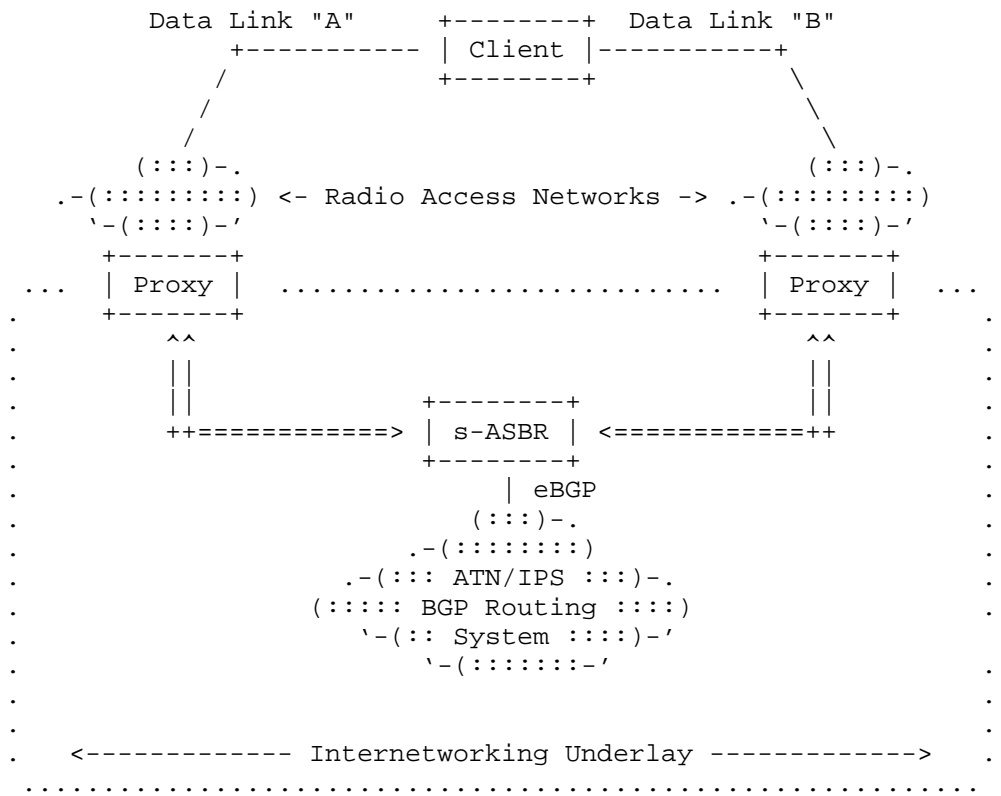


Figure 2: ATN/IPS RAN Architecture

When a Client logs into a RAN, it specifies a nearby s-ASBR that it has selected to connect to the ATN/IPS. The login process is brokered by a Proxy at the border of the RAN, which then conveys the connection request to the s-ASBR via tunneling across the Internetworking underlay. The s-ASBR then registers the address of the Proxy as the address for the Client, and the Proxy forwards the s-ASBR's reply to the Client. If the Client connects to multiple RANs, the s-ASBR will register the addresses of all Proxies as addresses through which the Client can be reached.

The s-ASBR represents all of its active Clients as MNP routes in the ATN/IPS BGP routing system. The s-ASBR's stub AS therefore consists of the set of all of its active Clients (i.e., the stub AS is a logical construct and not a physical construct). The s-ASBR injects the MNPs of its active Clients and withdraws the MNPs of its departed Clients via BGP updates to c-ASBRs. Since Clients are expected to remain associated with their current s-ASBR for extended periods, the level of MNP injections and withdrawals in the BGP routing system

will be on the order of the numbers of network joins, leaves and s-ASBR handovers for aircraft operations (see: Section 6). It is important to observe that fine-grained events such as Client mobility and Quality of Service (QoS) signaling are coordinated only by Proxies and the Client's current s-ASBRs, and do not involve other ASBRs in the routing system. In this way, intradomain routing changes within the stub AS are not propagated into the rest of the ATN/IPS BGP routing system.

5. ATN/IPS Route Optimization

ATN/IPS end systems will frequently need to communicate with correspondents associated with other s-ASBRs. In the BGP peering topology discussed in Section 3, this can initially only be accommodated by including multiple tunnel segments in the forwarding path. In many cases, it would be desirable to eliminate extraneous tunnel segments from this "dogleg" route so that packets can traverse a minimum number of tunneling hops across the Internetworking underlay. ATN/IPS end systems could therefore employ a route optimization service such as that discussed in [I-D.templin-aerolink].

A route optimization example is shown in Figure 3 and Figure 4 below. In the first figure, multiple tunneled segments between Proxys and ASBRs are necessary to convey packets between Clients associated with different s-ASBRs. In the second figure, the optimized route tunnels packets directly between Proxys without involving the ASBRs.

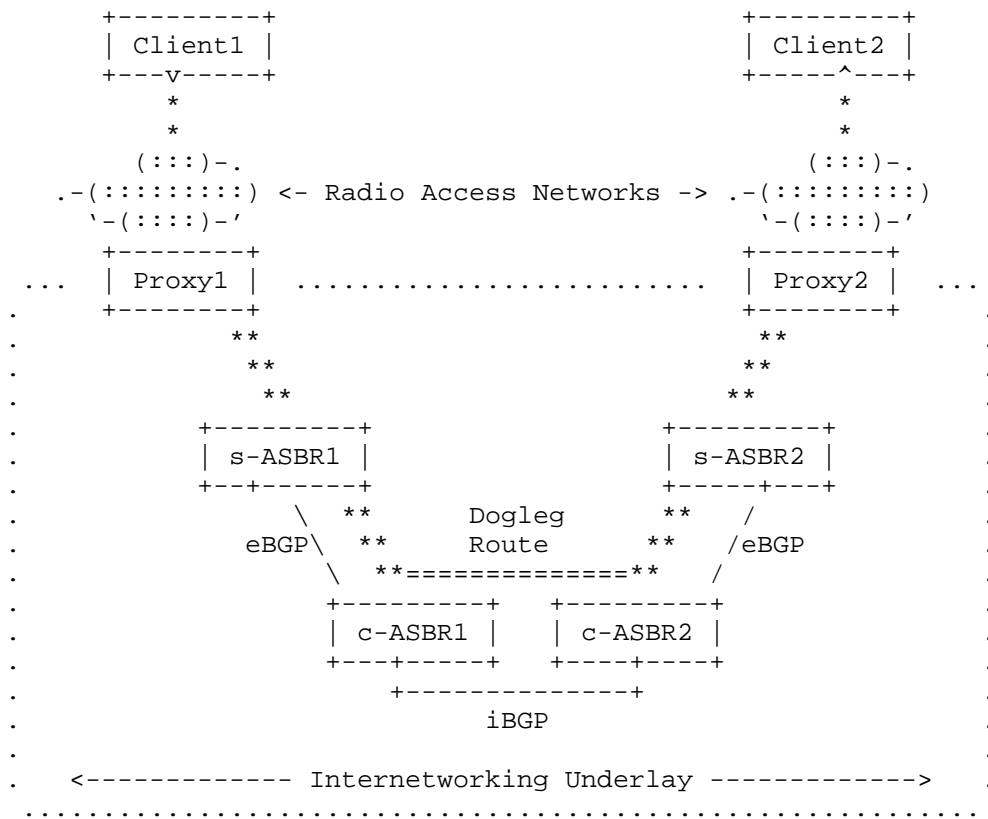


Figure 3: Dogleg Route Before Optimization

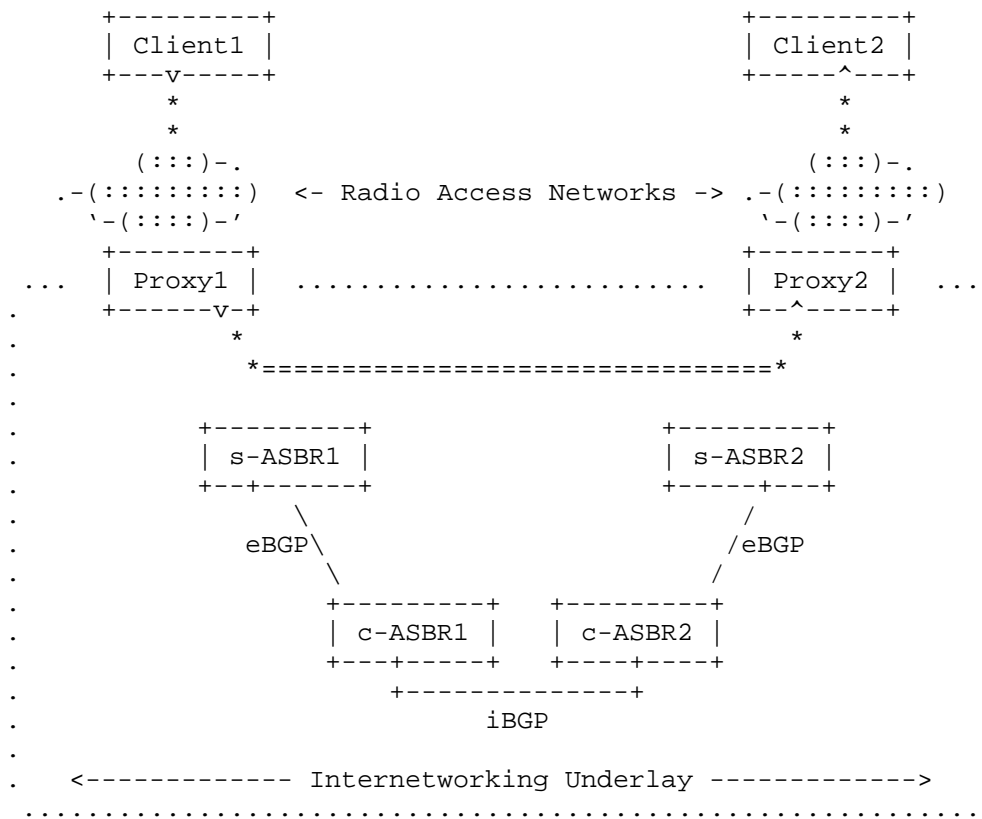


Figure 4: Optimized Route

6. BGP Protocol Considerations

The number of eBGP peering sessions that each c-ASBR must service is proportional to the number of s-ASBRs in the system. Network emulations with lightweight virtual machines have shown that a single c-ASBR can service at least 100 eBGP peerings from s-ASBRs that each advertise 10K MNP routes (i.e., 1M total). It is expected that robust c-ASBRs can service many more peerings than this - possibly by multiple orders of magnitude. But even assuming a conservative limit, the number of s-ASBRs could be increased by also increasing the number of c-ASBRs. Since c-ASBRs also peer with each other using iBGP, however, larger-scale c-ASBR deployments may need to employ an adjunct facility such as BGP Route Reflectors (RRs)[RFC4456].

The number of aircraft in operation at a given time worldwide is likely to be significantly less than 1M, but we will assume this number for a worst-case analysis. Assuming a worst-case average 1

hour flight profile from gate-to-gate with 10 service region transitions per flight, the entire system will need to service at most 10M BGP updates per hour (2778 updates per second). This number is within the realm of the peak BGP update messaging seen in the global public Internet today [BGP2]. Assuming a BGP update message size of 100 bytes (800bits), the total amount of BGP control message traffic to a single c-ASBR will be less than 2.5Mbps which is a nominal rate for modern data links.

Industry standard BGP routers provide configurable parameters with conservative default values. For example, the default hold time is 90 seconds, the default keepalive time is 1/3 of the hold time, and the default MinRouteAdvertisementInterval is 30 seconds for eBGP peers and 5 seconds for iBGP peers (see Section 10 of [RFC4271]). For the simple mobile routing system described herein, these parameters can and should be set to more aggressive values to support faster neighbor/link failure detection and faster routing protocol convergence times. For example, a hold time of 3 seconds and a MinRouteAdvertisementInterval of 0 seconds for both iBGP and eBGP.

Each c-ASBR will be using eBGP both in the ATN/IPS and the Internetworking Underlay with the ATN/IPS unicast IPv6 routes resolving over Internetworking Underlay routes. Consequently, c-ASBRs and potentially s-ASBRs will need to support separate local ASes for the two BGP routing domains and routing policy or assure routes are not propagated between the two BGP routing domains. From a conceptual and operational standpoint, the implementation should provide isolation between the two BGP routing domains (e.g., separate BGP instances).

7. Implementation Status

The BGP routing topology described in this document has been modeled in realistic network emulations showing that at least 1 million MNPs can be propagated to each c-ASBR even on lightweight virtual machines. No BGP routing protocol extensions need to be adopted.

8. IANA Considerations

This document does not introduce any IANA considerations.

9. Security Considerations

ATN/IPS ASBRs on the open Internet are susceptible to the same attack profiles as for any Internet nodes. For this reason, ASBRs should employ physical security and/or IP securing mechanisms such as IPsec [RFC4301], TLS [RFC5246], etc.

ATN/IPS ASBRs present targets for Distributed Denial of Service (DDoS) attacks. This concern is no different than for any node on the open Internet, where attackers could send spoofed packets to the node at high data rates. This can be mitigated by connecting ATN/IPS ASBRs over dedicated links with no connections to the Internet and/or when ASBR connections to the Internet are only permitted through well-managed firewalls.

ATN/IPS s-ASBRs should institute rate limits to protect low data rate aviation data links from receiving DDoS packet floods.

This document does not include any new specific requirements for mitigation of DDoS.

10. Acknowledgements

This work is aligned with the FAA as per the SE2025 contract number DTFAWA-15-D-00030.

This work is aligned with the NASA Safe Autonomous Systems Operation (SASO) program under NASA contract number NNA16BD84C.

This work is aligned with the Boeing Information Technology (BIT) MobileNet program.

11. References

11.1. Normative References

- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, DOI 10.17487/RFC4456, April 2006, <<https://www.rfc-editor.org/info/rfc4456>>.

- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

11.2. Informative References

- [BGP] Huston, G., "BGP in 2015, <http://potaroo.net>", January 2016.
- [BGP2] Huston, G., "BGP Instability Report, <http://bgpupdates.potaroo.net/instability/bgpupd.html>", May 2017.
- [CBB] Dul, A., "Global IP Network Mobility using Border Gateway Protocol (BGP), http://www.quark.net/docs/Global_IP_Network_Mobility_using_BGP.pdf", March 2006.
- [I-D.templin-aerolink] Templin, F., "Asymmetric Extended Route Optimization (AERO)", draft-templin-aerolink-82 (work in progress), May 2018.
- [ICAO] ICAO, I., "<http://www.icao.int/Pages/default.aspx>", February 2017.
- [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>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC6996] Mitchell, J., "Autonomous System (AS) Reservation for Private Use", BCP 6, RFC 6996, DOI 10.17487/RFC6996, July 2013, <<https://www.rfc-editor.org/info/rfc6996>>.

Authors' Addresses

Fred L. Templin (editor)
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org

Greg Saccone
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: gregory.t.saccone@boeing.com

Gaurav Dawra
LinkedIn
USA

Email: gdawra.ietf@gmail.com

Acee Lindem
Cisco Systems, Inc.
USA

Email: acee@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: February 17, 2019

F. Templin, Ed.
G. Saccone
Boeing Research & Technology
G. Dawra
LinkedIn
A. Lindem
V. Moreno
Cisco Systems, Inc.
August 16, 2018

A Simple BGP-based Mobile Routing System for the Aeronautical
Telecommunications Network
draft-templin-atn-bgp-08.txt

Abstract

The International Civil Aviation Organization (ICAO) is investigating mobile routing solutions for a worldwide Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS). The ATN/IPS will eventually replace existing communication services with an IPv6-based service supporting pervasive Air Traffic Management (ATM) for Air Traffic Controllers (ATC), Airline Operations Controllers (AOC), and all commercial aircraft worldwide. This informational document describes a simple and extensible mobile routing service based on industry-standard BGP to address the ATN/IPS requirements.

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 February 17, 2019.

Copyright Notice

Copyright (c) 2018 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. Terminology	5
3. ATN/IPS Routing System	6
4. ATN/IPS Radio Access Network (RAN) Model	9
5. ATN/IPS Route Optimization	11
6. BGP Protocol Considerations	13
7. Implementation Status	14
8. IANA Considerations	14
9. Security Considerations	14
10. Acknowledgements	15
11. References	15
11.1. Normative References	15
11.2. Informative References	16
Appendix A. Change Log	17
Authors' Addresses	17

1. Introduction

The worldwide Air Traffic Management (ATM) system today uses a service known as Aeronautical Telecommunications Network based on Open Systems Interconnection (ATN/OSI). The service is used to augment controller to pilot voice communications with rudimentary short text command and control messages. The service has seen successful deployment in a limited set of worldwide ATM domains.

The International Civil Aviation Organization [ICAO] is now undertaking the development of a next-generation replacement for ATN/OSI known as Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS). ATN/IPS will eventually provide an IPv6-based [RFC8200] service supporting pervasive ATM for Air Traffic Controllers (ATC), Airline Operations Controllers (AOC), and all

commercial aircraft worldwide. As part of the ATN/IPS undertaking, a new mobile routing service will be needed. This document presents an approach based on the Border Gateway Protocol (BGP) [RFC4271].

Aircraft communicate via wireless aviation data links that typically support much lower data rates than terrestrial wireless and wired-line communications. For example, some Very High Frequency (VHF)-based data links only support data rates on the order of 32Kbps and an emerging L-Band data link that is expected to play a key role in future aeronautical communications only supports rates on the order of 1Mbps. Although satellite data links can provide much higher data rates during optimal conditions, like any other aviation data link they are subject to errors, delay, disruption, signal intermittence, degradation due to atmospheric conditions, etc. The well-connected ground domain ATN/IPS network should therefore treat each safety-of-flight critical packet produced by (or destined to) an aircraft as a precious commodity and strive for an optimized service that provides the highest possible degree of reliability.

The ATN/IPS is an IPv6-based overlay network that assumes a worldwide connected Internetworking underlay for carrying tunneled ATM communications. The Internetworking underlay could be manifested as a private collection of long-haul backbone links (e.g., fiber optics, copper, SATCOM, etc.) interconnected by high-performance networking gear such as bridges, switches, and routers. Such a private network would need to connect all ATN/IPS participants worldwide, and could therefore present a considerable cost for a large-scale deployment of new infrastructure. Alternatively, the ATN/IPS could be deployed as a secured overlay over the existing global public Internet. For example, ATN/IPS nodes could be deployed as part of an SD-WAN or an MPLS-WAN that rides over the public Internet via secured tunnels. Further details of the Internetworking underlay design are out of scope for this document.

The ATN/IPS further assumes that each aircraft will receive an IPv6 Mobile Network Prefix (MNP) that accompanies the aircraft wherever it travels. ICAO is further proposing to assign each aircraft an entire /56 MNP for numbering its on-board networks. ATCs and AOCs will likewise receive IPv6 prefixes, but they would typically appear in static (not mobile) deployments such as air traffic control towers, airline headquarters, etc. Throughout the rest of this document, we therefore use the term "MNP" when discussing an IPv6 prefix that is delegated to any ATN/IPS end system, including ATCs, AOCs, and aircraft. We also use the term Mobility Service Prefix (MSP) to refer to an aggregated prefix assigned to the ATN/IPS by an Internet assigned numbers authority, and from which all MNPs are delegated (e.g., up to 2^{32} IPv6 /56 MNPs could be delegated from an IPv6 /24 MSP).

Connexion By Boeing [CBB] was an early aviation mobile routing service based on dynamic updates in the global public Internet BGP routing system. Practical experience with the approach has shown that frequent injections and withdrawals of MNPs in the Internet routing system can result in excessive BGP update messaging, slow routing table convergence times, and extended outages when no route is available. This is due to both conservative default BGP protocol timing parameters (see Section 6) and the complex peering interconnections of BGP routers within the global Internet infrastructure. The situation is further exacerbated by frequent aircraft mobility events that each result in BGP updates that must be propagated to all BGP routers in the Internet that carry a full routing table.

We therefore consider an approach using a BGP overlay network routing system where a private BGP routing protocol instance is maintained between ATN/IPS Autonomous System (AS) Border Routers (ASBRs). The private BGP instance does not interact with the native BGP routing system in the connected Internetworking underlay, and BGP updates are unidirectional from "stub" ASBRs (s-ASBRs) to a small set of "core" ASBRs (c-ASBRs) in a hub-and-spokes topology. No extensions to the BGP protocol are necessary.

The s-ASBRs for each stub AS connect to a small number of c-ASBRs via dedicated high speed links and/or tunnels across the Internetworking underlay using industry-standard encapsulations (e.g., Generic Routing Encapsulation (GRE) [RFC2784], IPsec [RFC4301], etc.). In particular, tunneling must be used when neighboring ASBRs are separated by many Internetworking underlay hops.

The s-ASBRs engage in external BGP (eBGP) peerings with their respective c-ASBRs, and only maintain routing table entries for the MNPs currently active within the stub AS. The s-ASBRs send BGP updates for MNP injections or withdrawals to c-ASBRs but do not receive any BGP updates from c-ASBRs. Instead, the s-ASBRs maintain default routes with their c-ASBRs as the next hop, and therefore hold only partial topology information.

The c-ASBRs connect to other c-ASBRs using internal BGP (iBGP) peerings over which they collaboratively maintain a full routing table for all active MNPs currently in service. Therefore, only the c-ASBRs maintain a full BGP routing table and never send any BGP updates to s-ASBRs. This simple routing model therefore greatly reduces the number of BGP updates that need to be synchronized among peers, and the number is reduced further still when intradomain routing changes within stub ASes are processed within the AS instead of being propagated to the core. BGP Route Reflectors (RRs) [RFC4456] can also be used to support increased scaling properties.

The remainder of this document discusses the proposed BGP-based ATN/IPS mobile routing service.

2. Terminology

The terms Autonomous System (AS) and Autonomous System Border Router (ASBR) are the same as defined in [RFC4271].

The following terms are defined for the purposes of this document:

Air Traffic Management (ATM)

The worldwide service for coordinating safe aviation operations.

Air Traffic Controller (ATC)

A government agent responsible for coordinating with aircraft within a defined operational region via voice and/or data Command and Control messaging.

Airline Operations Controller (AOC)

An airline agent responsible for tracking and coordinating with aircraft within their fleet.

Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS)

A future aviation network for ATCs and AOCs to coordinate with all aircraft operating worldwide. The ATN/IPS will be an IPv6-based overlay network service that connects access networks via tunneling over an Internetworking underlay.

Internetworking underlay A connected wide-area network that supports overlay network tunneling and connects Radio Access Networks to the rest of the ATN/IPS.

Radio Access Network (RAN)

An aviation radio data link service provider's network, including radio transmitters and receivers as well as supporting ground-domain infrastructure needed to convey a customer's data packets to the outside world. The term RAN is intended in the same spirit as for cellular operator networks and other radio-based Internet service provider networks. For simplicity, we also use the term RAN to refer to ground-domain networks that connect AOCs and ATCs without any aviation radio communications.

Core Autonomous System Border Router (c-ASBR) A BGP router located in the hub of the ATN/IPS hub-and-spokes overlay network topology.

Core Autonomous System The "hub" autonomous system maintained by all c-ASBRs.

Stub Autonomous System Border Router (s-ASBR) A BGP router configured as a spoke in the ATN/IPS hub-and-spokes overlay network topology.

Stub Autonomous System A logical grouping that includes all Clients currently associated with a given s-ASBR.

Client An ATC, AOC or aircraft that connects to the ATN/IPS as a leaf node. The Client could be a singleton host, or a router that connects a mobile network.

Proxy A node at the edge of a RAN that acts as an intermediary between Clients and s-ASBRs. From the Client's perspective, the Proxy presents the appearance that the Client is communicating directly with the s-ASBR. From the s-ASBR's perspective, the Proxy presents the appearance that the s-ASBR is communicating directly with the Client.

Mobile Network Prefix (MNP) An IPv6 prefix that is delegated to any ATN/IPS end system, including ATCs, AOCs, and aircraft.

Mobility Service Prefix (MSP) An aggregated prefix assigned to the ATN/IPS by an Internet assigned numbers authority, and from which all MNPs are delegated (e.g., up to 2^{32} IPv6 /56 MNPs could be delegated from a /24 MSP).

3. ATN/IPS Routing System

The proposed ATN/IPS routing system comprises a private BGP instance coordinated in an overlay network via tunnels between neighboring ASBRs over the Internetworking underlay. The overlay does not interact with the native BGP routing system in the connected underlying Internetwork, and each c-ASBR advertises only a small and unchanging set of MSPs into the Internetworking underlay routing system instead of the full dynamically changing set of MNPs. (For example, when the Internetworking underlay is the global public Internet the c-ASBRs advertise the MSPs in the public BGP Internet routing system.)

In a reference deployment, one or more s-ASBRs connect each stub AS to the overlay using a shared stub AS Number (ASN). Each s-ASBR further uses eBGP to peer with one or more c-ASBRs. All c-ASBRs are members of the same core AS, and use a shared core ASN. Globally-unique public ASNs could be assigned, e.g., either according to the standard 16-bit ASN format or the 32-bit ASN scheme defined in [RFC6793].

The c-ASBRs use iBGP to maintain a synchronized consistent view of all active MNPs currently in service. Figure 1 below represents the reference deployment. (Note that the figure shows details for only two s-ASBRs (s-ASBR1 and s-ASBR2) due to space constraints, but the other s-ASBRs should be understood to have similar Stub AS, MNP and eBGP peering arrangements.) The solution described in this document is flexible enough to extend to these topologies.

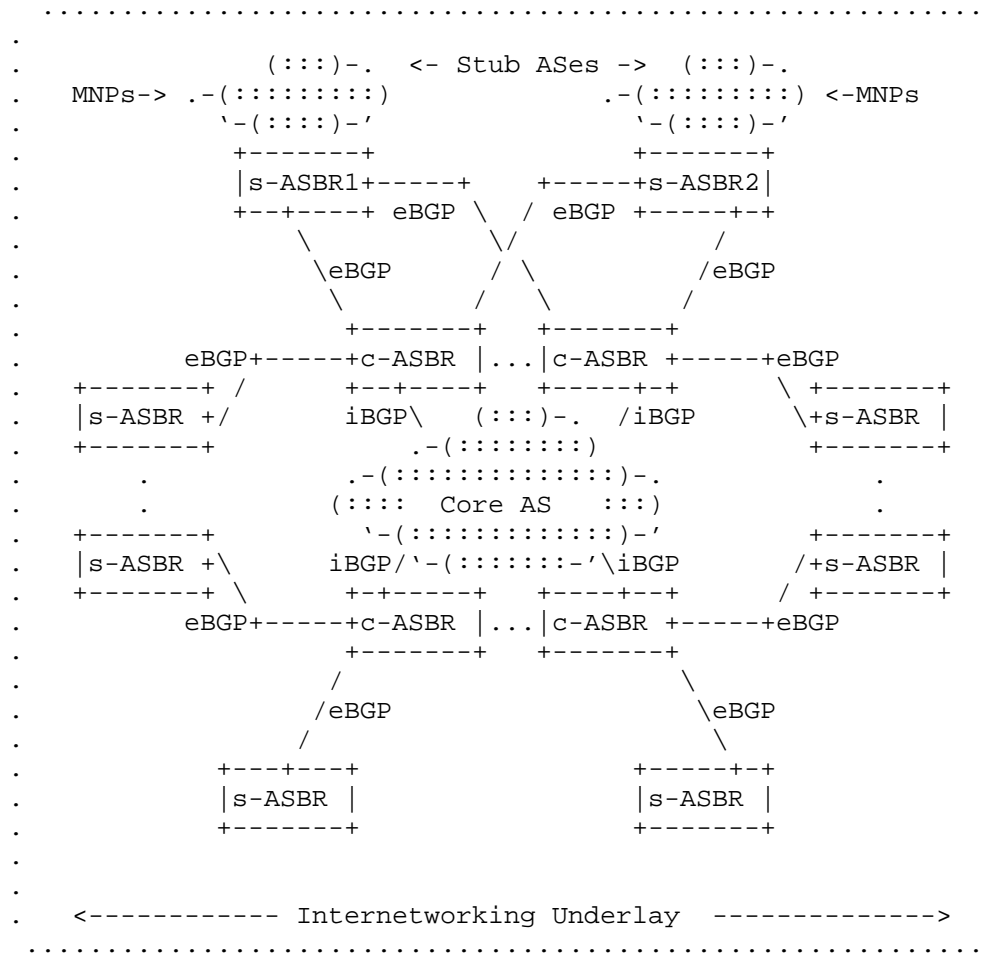


Figure 1: Reference Deployment

In the reference deployment, each s-ASBR maintains routes for active MNPs that currently belong to its stub AS. In response to "Inter-domain" mobility events, each s-ASBR will dynamically announces new MNPs and withdraws departed MNPs in its eBGP updates to c-ASBRs.

Since ATN/IPS end systems are expected to remain within the same stub AS for extended timeframes, however, intra-domain mobility events (such as an aircraft handing off between cell towers) are handled within the stub AS instead of being propagated as inter-domain eBGP updates.

Each c-ASBR configures a black-hole route for each of its MSPs. By black-holing the MSPs, the c-ASBR will maintain forwarding table entries only for the MNPs that are currently active, and packets destined to all other MNPs will correctly incur ICMPv6 Destination Unreachable messages [RFC4443] due to the black hole route. (This is the same behavior as for ordinary BGP routers in the Internet when they receive packets for which there is no route available.) The c-ASBRs do not send eBGP updates for MNPs to s-ASBRs, but instead originate a default route. In this way, s-ASBRs have only partial topology knowledge (i.e., they know only about the active MNPs currently within their stub ASes) and they forward all other packets to c-ASBRs which have full topology knowledge.

Scaling properties of this ATN/IPS routing system are limited by the number of BGP routes that can be carried by the c-ASBRs. A 2015 study showed that BGP routers in the global public Internet at that time carried more than 500K routes with linear growth and no signs of router resource exhaustion [BGP]. A more recent network emulation study also showed that a single c-ASBR can accommodate at least 1M dynamically changing BGP routes even on a lightweight virtual machine. Commercially-available high-performance dedicated router hardware can support many millions of routes.

Therefore, assuming each c-ASBR can carry 1M or more routes, this means that at least 1M ATN/IPS end system MNPs can be serviced by a single set of c-ASBRs and that number could be further increased by using RRs and/or more powerful routers. Another means of increasing scale would be to assign a different set of c-ASBRs for each set of MSPs. In that case, each s-ASBR still peers with one or more c-ASBRs from each set of c-ASBRs, but the s-ASBR institutes route filters so that it only sends BGP updates to the specific set of c-ASBRs that aggregate the MSP. In this way, each set of c-ASBRs maintains separate routing and forwarding tables so that scaling is distributed across multiple c-ASBR sets instead of concentrated in a single c-ASBR set. For example, a first c-ASBR set could aggregate an MSP segment A::/32, a second set could aggregate B::/32, a third could aggregate C::/32, etc. The union of all MSP segments would then constitute the collective MSP(s) for the entire ATN/IPS.

In this way, each set of c-ASBRs services a specific set of MSPs that they inject into the Internetworking underlay native routing system, and each s-ASBR configures MSP-specific routes that list the correct

set of c-ASBRs as next hops. This design also allows for natural incremental deployment, and can support initial medium-scale deployments followed by dynamic deployment of additional ATN/IPS infrastructure elements without disturbing the already-deployed base. For example, a few more c-ASBRs could be added if the MNP service demand ever outgrows the initial deployment.

4. ATN/IPS Radio Access Network (RAN) Model

Radio Access Networks (RANs) connect end system Clients such as aircraft, ATCs, AOCs etc. to the ATN/IPS routing system. Clients may connect to multiple RANs at once, for example, when they have both satellite and cellular data links activated simultaneously. Clients may further move between RANs in a manner that is perceived as a network layer mobility event. Clients could therefore employ a multilink/mobility routing service such as that discussed in [I-D.templin-aerolink].

Clients register all of their active data link connections with their serving s-ASBRs as discussed in Section 3. Clients may connect to s-ASBRs either directly, or via a Proxy at the edge of the RAN.

Figure 2 shows the ATN/IPS RAN model where Clients connect to RANs via aviation data links. Clients register their RAN addresses with a nearby s-ASBR, where the registration process may be brokered by a Proxy at the edge of the RAN.

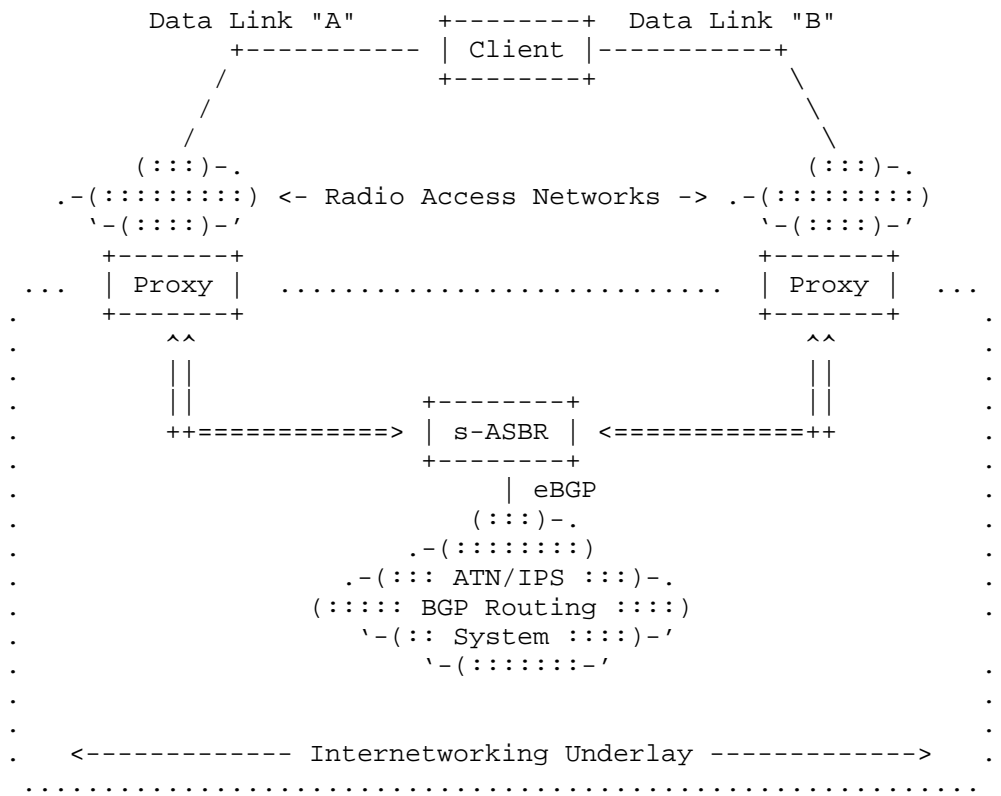


Figure 2: ATN/IPS RAN Architecture

When a Client logs into a RAN, it specifies a nearby s-ASBR that it has selected to connect to the ATN/IPS. The login process is brokered by a Proxy at the border of the RAN, which then conveys the connection request to the s-ASBR via tunneling across the Internetworking underlay. The s-ASBR then registers the address of the Proxy as the address for the Client, and the Proxy forwards the s-ASBR's reply to the Client. If the Client connects to multiple RANs, the s-ASBR will register the addresses of all Proxies as addresses through which the Client can be reached.

The s-ASBR represents all of its active Clients as MNP routes in the ATN/IPS BGP routing system. The s-ASBR's stub AS therefore consists of the set of all of its active Clients (i.e., the stub AS is a logical construct and not a physical construct). The s-ASBR injects the MNPs of its active Clients and withdraws the MNPs of its departed Clients via BGP updates to c-ASBRs. Since Clients are expected to remain associated with their current s-ASBR for extended periods, the level of MNP injections and withdrawals in the BGP routing system

will be on the order of the numbers of network joins, leaves and s-ASBR handovers for aircraft operations (see: Section 6). It is important to observe that fine-grained events such as Client mobility and Quality of Service (QoS) signaling are coordinated only by Proxies and the Client's current s-ASBRs, and do not involve other ASBRs in the routing system. In this way, intradomain routing changes within the stub AS are not propagated into the rest of the ATN/IPS BGP routing system.

5. ATN/IPS Route Optimization

ATN/IPS end systems will frequently need to communicate with correspondents associated with other s-ASBRs. In the BGP peering topology discussed in Section 3, this can initially only be accommodated by including multiple tunnel segments in the forwarding path. In many cases, it would be desirable to eliminate extraneous tunnel segments from this "dogleg" route so that packets can traverse a minimum number of tunneling hops across the Internetworking underlay. ATN/IPS end systems could therefore employ a route optimization service such as that discussed in [I-D.templin-aerolink].

A route optimization example is shown in Figure 3 and Figure 4 below. In the first figure, multiple tunneled segments between Proxys and ASBRs are necessary to convey packets between Clients associated with different s-ASBRs. In the second figure, the optimized route tunnels packets directly between Proxys without involving the ASBRs.

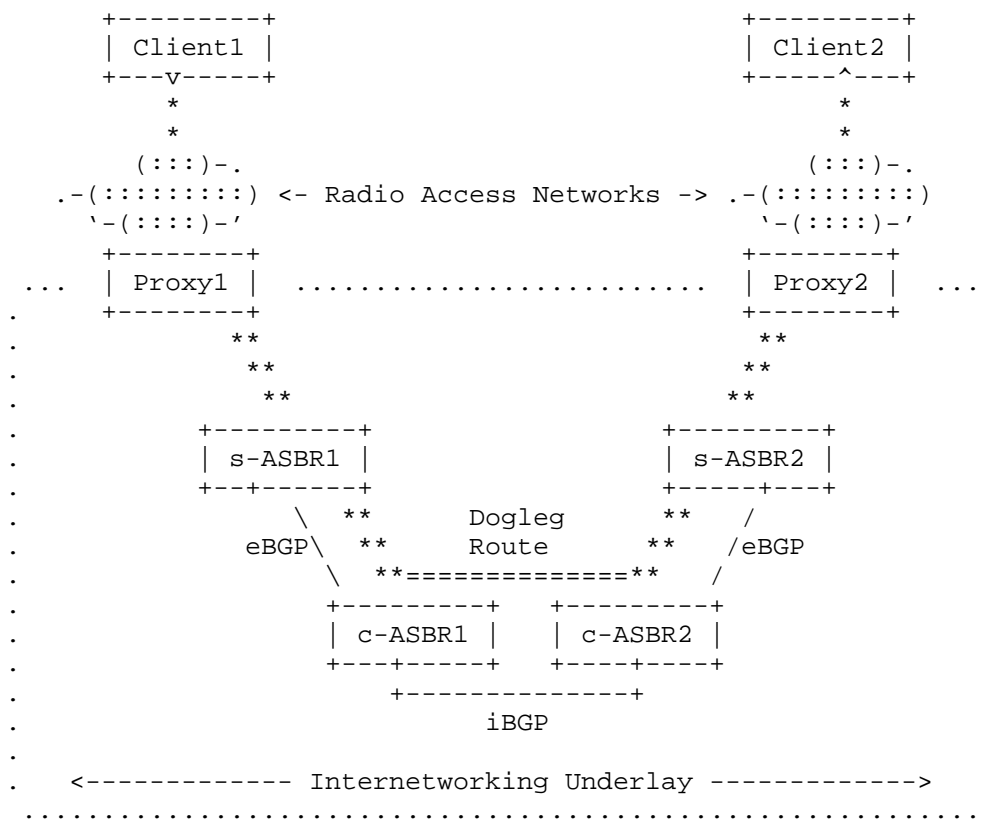


Figure 3: Dogleg Route Before Optimization

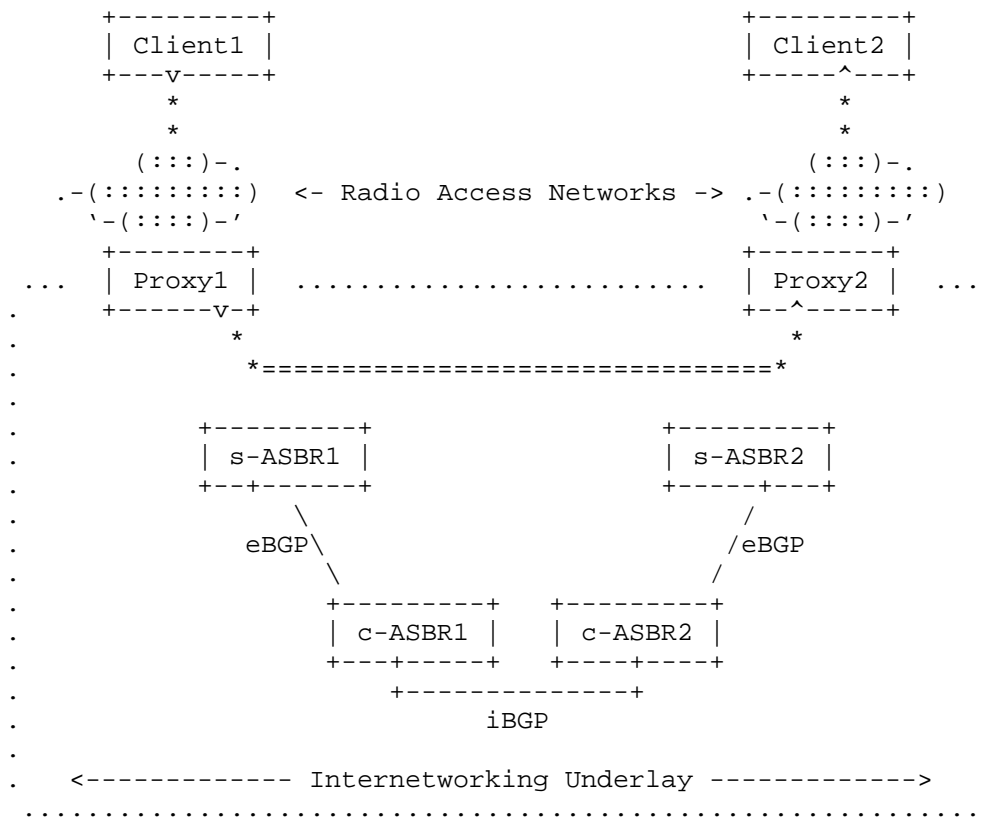


Figure 4: Optimized Route

6. BGP Protocol Considerations

The number of eBGP peering sessions that each c-ASBR must service is proportional to the number of s-ASBRs in the system. Network emulations with lightweight virtual machines have shown that a single c-ASBR can service at least 100 eBGP peerings from s-ASBRs that each advertise 10K MNP routes (i.e., 1M total). It is expected that robust c-ASBRs can service many more peerings than this - possibly by multiple orders of magnitude. But even assuming a conservative limit, the number of s-ASBRs could be increased by also increasing the number of c-ASBRs. Since c-ASBRs also peer with each other using iBGP, however, larger-scale c-ASBR deployments may need to employ an adjunct facility such as BGP Route Reflectors (RRs)[RFC4456].

The number of aircraft in operation at a given time worldwide is likely to be significantly less than 1M, but we will assume this number for a worst-case analysis. Assuming a worst-case average 1

hour flight profile from gate-to-gate with 10 service region transitions per flight, the entire system will need to service at most 10M BGP updates per hour (2778 updates per second). This number is within the realm of the peak BGP update messaging seen in the global public Internet today [BGP2]. Assuming a BGP update message size of 100 bytes (800bits), the total amount of BGP control message traffic to a single c-ASBR will be less than 2.5Mbps which is a nominal rate for modern data links.

Industry standard BGP routers provide configurable parameters with conservative default values. For example, the default hold time is 90 seconds, the default keepalive time is 1/3 of the hold time, and the default MinRouteAdvertisementInterval is 30 seconds for eBGP peers and 5 seconds for iBGP peers (see Section 10 of [RFC4271]). For the simple mobile routing system described herein, these parameters can and should be set to more aggressive values to support faster neighbor/link failure detection and faster routing protocol convergence times. For example, a hold time of 3 seconds and a MinRouteAdvertisementInterval of 0 seconds for both iBGP and eBGP.

Each c-ASBR will be using eBGP both in the ATN/IPS and the Internetworking Underlay with the ATN/IPS unicast IPv6 routes resolving over Internetworking Underlay routes. Consequently, c-ASBRs and potentially s-ASBRs will need to support separate local ASes for the two BGP routing domains and routing policy or assure routes are not propagated between the two BGP routing domains. From a conceptual and operational standpoint, the implementation should provide isolation between the two BGP routing domains (e.g., separate BGP instances).

7. Implementation Status

The BGP routing topology described in this document has been modeled in realistic network emulations showing that at least 1 million MNPs can be propagated to each c-ASBR even on lightweight virtual machines. No BGP routing protocol extensions need to be adopted.

8. IANA Considerations

This document does not introduce any IANA considerations.

9. Security Considerations

ATN/IPS ASBRs on the open Internet are susceptible to the same attack profiles as for any Internet nodes. For this reason, ASBRs should employ physical security and/or IP securing mechanisms such as IPsec [RFC4301], TLS [RFC5246], etc.

ATN/IPS ASBRs present targets for Distributed Denial of Service (DDoS) attacks. This concern is no different than for any node on the open Internet, where attackers could send spoofed packets to the node at high data rates. This can be mitigated by connecting ATN/IPS ASBRs over dedicated links with no connections to the Internet and/or when ASBR connections to the Internet are only permitted through well-managed firewalls.

ATN/IPS s-ASBRs should institute rate limits to protect low data rate aviation data links from receiving DDoS packet floods.

This document does not include any new specific requirements for mitigation of DDoS.

10. Acknowledgements

This work is aligned with the FAA as per the SE2025 contract number DTFAWA-15-D-00030.

This work is aligned with the NASA Safe Autonomous Systems Operation (SASO) program under NASA contract number NNA16BD84C.

This work is aligned with the Boeing Information Technology (BIT) MobileNet program.

11. References

11.1. Normative References

- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, DOI 10.17487/RFC4456, April 2006, <<https://www.rfc-editor.org/info/rfc4456>>.

- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

11.2. Informative References

- [BGP] Huston, G., "BGP in 2015, <http://potaroo.net>", January 2016.
- [BGP2] Huston, G., "BGP Instability Report, <http://bgpupdates.potaroo.net/instability/bgpupd.html>", May 2017.
- [CBB] Dul, A., "Global IP Network Mobility using Border Gateway Protocol (BGP), http://www.quark.net/docs/Global_IP_Network_Mobility_using_BGP.pdf", March 2006.
- [I-D.templin-aerolink] Templin, F., "Asymmetric Extended Route Optimization (AERO)", draft-templin-aerolink-82 (work in progress), May 2018.
- [ICAO] ICAO, I., "<http://www.icao.int/Pages/default.aspx>", February 2017.
- [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>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC6793] Vohra, Q. and E. Chen, "BGP Support for Four-Octet Autonomous System (AS) Number Space", RFC 6793, DOI 10.17487/RFC6793, December 2012, <<https://www.rfc-editor.org/info/rfc6793>>.

Appendix A. Change Log

<< RFC Editor - remove prior to publication >>

Changes from -07 to -08:

- o Removed suggestion to use private ASNs
- o Ran spelling checker and corrected errors
- o Re-worked Section 3 final two paragraphs on scaling
- o Stated Internetwork underlay as being out of scope for this document

Authors' Addresses

Fred L. Templin (editor)
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org

Greg Saccone
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: gregory.t.saccone@boeing.com

Gaurav Dawra
LinkedIn
USA

Email: gdawra.ietf@gmail.com

Acee Lindem
Cisco Systems, Inc.
USA

Email: acee@cisco.com

Victor Moreno
Cisco Systems, Inc.
USA

Email: vimoreno@cisco.com