LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: September 6, 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

March 5, 2018

LISP YANG Model
draft-ietf-lisp-yang-07

Abstract

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

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 6, 2018.

Table of Contents

1. Introduction

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

2. LISP Module

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

2.1. Module Structure

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

```
       |              |     |     +--rw mask-length?   uint8
```

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

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

2.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp@2018-03-05.yang"
module ietf-lisp {
  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp";
  prefix lisp;
  import ietf-interfaces {
    prefix if;
  }
  import ietf-lisp-address-types {
    prefix lcaf;
  }
  import ietf-yang-types {
    prefix yang;
  }
  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.
    ";
  // RFC Ed.: replace XXXX with actual RFC number and remove
  // this note
  reference "RFC XXXX";

  revision 2018-03-05 {
    description
      "Initial revision.";
    reference
      "https://tools.ietf.org/html/rfc6830";
  }
  identity lisp-role {
    description
      "LISP router role.";
  }
  identity itr {
    base lisp-role;
    description
      "LISP ITR.";
  }
  identity pitr {
    base lisp-role;
    description
      "LISP PITR.";
  }
  identity etr {
    base lisp-role;
    description
      "LISP ETR.";
  }
  identity petr {
    base lisp-role;
    description
```

```
        "LISP PETR.";
    }
    identity mapping-system {
      description
        "Mapping System interface";
    }
    identity single-node-mapping-system {
      base mapping-system;
      description
        "logically singular Map Server";
    }
    typedef mapping-system-ref {
      type identityref {
        base mapping-system;
      }
      description
        "Mapping System reference";
    }

    typedef lisp-role-ref {
      type identityref {
        base lisp-role;
      }
      description
        "LISP role reference";
    }
    typedef map-reply-action {
      type enumeration {
        enum no-action {
          value 0;
          description
            "Mapping is kept alive and no encapsulation occurs.";
        }
        enum natively-forward {
          value 1;
          description
            "Matching packets are not encapsulated or dropped but
             natively forwarded.";
        }
        enum send-map-request {
          value 2;
          description
            "Matching packets invoke Map-Requests.";
        }
        enum drop {
          value 3;
          description
            "Matching packets are dropped.";
```

```
        }
      }
      description
        "Defines the lisp map-cache ACT type";
      reference "https://tools.ietf.org/html/rfc6830#section-6.1.4";
    }
    typedef eid-id {
      type string;
      description
        "Type encoding of lisp-addresses to be generally used in EID
         keyed lists.";
    }
    typedef auth-key-type {
      type enumeration {
        enum none {
          value 0;
          description
            "No authentication.";
        }
        enum hmac-sha-1-96 {
          value 1;
          description
            "HMAC-SHA-1-96 (RFC2404) authentication is used.";
        }
        enum hmac-sha-256-128 {
          value 2;
          description
            "HMAC-SHA-256-128 (RFC4868) authentication is used.";
        }
      }
      description
        "Enumeration of the authentication mechanisms supported by
         LISP.";
      reference
        "https://tools.ietf.org/html/rfc6830#section-6.1.6";
    }
    typedef xtr-id-type {
      type binary {
        length "16";
      }
      description
      "128 bit xTR identifier.";
    }

    grouping locator-properties {
      description
        "Properties of a RLOC";
      leaf priority {
```

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

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

    }

    grouping local-locators-grouping {
      description
```

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

    grouping mapping {
      description
        "Group that defines a LISP mapping.";
      container eid {
        uses lcaf:lisp-address;
        description
          "End-host Identifier (EID) to be mapped to a list of
           locators";
      }
      leaf time-to-live {
        type uint32;
        units minutes;
        description
          "Mapping validity period in minutes.";
      }
      leaf creation-time {
        type yang:date-and-time;
        description
          "Time when the mapping was created.";
      }
      leaf authoritative {
        type bits {
          bit A {
            description
              "Authoritative bit.";
          }
        }
        description
          "Bit that indicates if mapping comes from an
           authoritative source.";
      }
      leaf static {
        type boolean;
        default "false";
```

```
        description
          "This leaf should be true if the mapping is static.";
      }
      choice locator-list {
        description
          "list of locartors are either negative, or positive.";
        case negative-mapping {
          leaf map-reply-action {
            type map-reply-action;
            description
              "Forwarding action for a negative mapping.";
          }
        }
        case positive-mapping {
          container rlocs {
            uses locators-grouping;
            description
              "List of locators for a positive mapping.";
          }
        }
      }
    }

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

```
          }
        }
      }
    }

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

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

      container lisp-router-instances {
        description
          "Different LISP routers instantiated in the device";
        list lisp-router-instance {
          key "lisp-router-instance-id";
```

```
          description
            "Each entry contains parameters for a LISP router.";
          leaf lisp-router-instance-id {
            type int32;
            description
              "Arbitrary lisp-router id.";
          }
          list lisp-role {
            key lisp-role-type;
            description
              "List of lisp device roles such as MS, MR, ITR,
              PITR, ETR or PETR.";
            leaf lisp-role-type {
              type lisp-role-ref;
              description
                "The type of LISP device - identity derived from the
                 'lisp-device' base identity.";
            }
          }
          container lisp-router-id {
            when "../lisp-role/lisp-role-type = 'itr' or
                  ../lisp-role/lisp-role-type = '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";
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

3. LISP-ITR Module

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

3.1. Module Structure

```
module: ietf-lisp-itr
  augment /lisp:lisp/lisp:lisp-router-instances/lisp:lisp-router-instance:
    +--rw itr!
       +--rw rloc-probing!
       |  +--rw interval?           uint16
       |  +--rw retries?            uint8
       |  +--rw retries-interval?   uint16
       +--rw itr-rlocs?       -> /lisp:lisp/locator-sets/locator-set/locator-
set-name
       +--rw map-resolvers
       |  +--rw map-resolver*   inet:ip-address
       +--rw proxy-etrs
       |  +--rw proxy-etr-address*   inet:ip-address
       +--rw map-cache
          +--rw virtual-network* [vni]
             +--rw vni         lcaf:instance-id-type
             +--rw mappings
                +--rw mapping* [id]
                   +--rw id                  eid-id
                   +--rw eid
                   |  +--rw address-type             lisp-address-family-ref
                   |  +--rw virtual-network-id?      instance-id-type
                   |  +--rw (address)?
                   |     +--:(no-address)
                   |     |  +--rw no-address?              empty
                   |     +--:(ipv4)
                   |     |  +--rw ipv4?                    inet:ipv4-address
                   |     +--:(ipv4-prefix)
                   |     |  +--rw ipv4-prefix?             inet:ipv4-prefix
                   |     +--:(ipv6)
                   |     |  +--rw ipv6?                    inet:ipv6-address
                   |     +--:(ipv6-prefix)
                   |     |  +--rw ipv6-prefix?             inet:ipv6-prefix
                   |     +--:(mac)
                   |     |  +--rw mac?                     yang:mac-address
                   |     +--:(distinguished-name)
                   |     |  +--rw distinguished-name?      distinguished-name-
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
                    +--rw creation-time?      yang:date-and-time
                    +--rw authoritative?      bits
                    +--rw static?             boolean
                    +--rw (locator-list)?
                       +--:(negative-mapping)
                       |  +--rw map-reply-action?   map-reply-action
                       +--:(positive-mapping)
                          +--rw rlocs
                             +--rw locator* [id]
                                +--rw id                    string
                                +--rw locator-address
                                |  +--rw address-type             lisp-address-
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@2018-03-05.yang"
module ietf-lisp-itr {
  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-itr";
  prefix lisp-itr;
  import ietf-lisp {
    prefix lisp;
  }
  import ietf-inet-types {
    prefix inet;
  }
  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "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.
    ";
  // RFC Ed.: replace XXXX with actual RFC number and remove
  // this note
  reference "RFC XXXX";

  revision 2018-03-05 {
    description
      "Initial revision.";
    reference
      "https://tools.ietf.org/html/rfc6830";
  }
  augment "/lisp:lisp/lisp:lisp-router-instances/lisp:lisp-router-instance" {
    when "lisp:lisp-role/lisp:lisp-role-type = 'lisp:itr' or
          lisp:lisp-role/lisp:lisp-role-type = 'lisp:pitr'" {
      description
        "Augment is valid when LISP role type is ITR or PITR.";
    }
    description
      "This augments LISP devices list with (P)ITR specific
       parameters.";
    container itr {
      presence "LISP (P)ITR operation enabled";
      description
        "ITR parameters";
      container rloc-probing {
        presence "RLOC probing active";
        description
          "RLOC-probing parameters";
        leaf interval {
          type uint16;
          units "seconds";
          description
            "Interval in seconds for resending the probes";
        }
        leaf retries {
          type uint8;
          description
            "Number of retries for sending the probes";
        }
        leaf retries-interval {
          type uint16;
          units "seconds";
          description
            "Interval in seconds between retries when sending probes.
             The action taken if all retries fail to receive is
             impementation specific.";
```

```
        }
      }
      leaf itr-rlocs {
        type leafref {
          path "/lisp:lisp/lisp:locator-sets/lisp:locator-set/"
          + "lisp:locator-set-name";
        }
        description
          "Reference to a locator set that the (P)ITR includes in
           Map-Requests";
      }
      container map-resolvers {
        description
          "Map-Resolvers that the (P)ITR uses.";
        leaf-list map-resolver {
          type inet:ip-address;
          min-elements 1;
          description
            "Each Map-Resolver within the list of Map-Resolvers.";
        }
      }
      container proxy-etrs {
        when "../../lisp:lisp-role/lisp:lisp-role-type = 'lisp:itr'" {
          description
            "Container exists only when LISP role type is ITR";
        }
        description
          "Proxy ETRs that the ITR uses.";
        leaf-list proxy-etr-address{
          type inet:ip-address;
          description
            "Proxy ETR RLOC address.";
        }
      }
      container map-cache{
        uses lisp:mappings;
        description
          "EID to RLOCs mappings cache.";
      }
    }
  }
}
<CODE ENDS>
```

4. LISP-ETR Module

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

4.1. Module Structure

```
module: ietf-lisp-etr
  augment /lisp:lisp/lisp:lisp-router-instances/lisp:lisp-router-instance:
    +--rw etr!
       +--rw map-servers
       |  +--rw map-server* [ms-address]
       |     +--rw ms-address       inet:ip-address
       |     +--rw auth-key?        string
       |     +--rw auth-key-type?   lisp:auth-key-type
       +--rw local-eids
          +--rw virtual-network* [vni]
             +--rw vni     lcaf:instance-id-type
             +--rw eids
                +--rw local-eid* [id]
                   +--rw id                        lisp:eid-id
                   +--rw eid-address
                   |  +--rw address-type             lisp-address-family-ref
                   |  +--rw virtual-network-id?      instance-id-type
                   |  +--rw (address)?
                   |     +--:(no-address)
                   |     |  +--rw no-address?              empty
                   |     +--:(ipv4)
                   |     |  +--rw ipv4?                    inet:ipv4-address
                   |     +--:(ipv4-prefix)
                   |     |  +--rw ipv4-prefix?             inet:ipv4-prefix
                   |     +--:(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?                    -> /lisp:lisp/locator-sets/
locator-set/locator-set-name
                    +--rw record-ttl?               uint32
                    +--rw want-map-notify?          boolean
                    +--rw proxy-reply?              boolean
                    +--rw registration-interval?    uint16
```

4.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp-etr@2018-03-05.yang"
module ietf-lisp-etr {
  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-etr";
  prefix lisp-etr;
  import ietf-lisp {
    prefix lisp;
  }
  import ietf-lisp-address-types {
    prefix lcaf;
  }
  import ietf-inet-types {
    prefix inet;
  }
  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.
```

```
     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 2018-03-05 {
    description
      "Initial revision.";
    reference
      "https://tools.ietf.org/html/rfc6830";
  }
  augment "/lisp:lisp/lisp:lisp-router-instances/lisp:lisp-router-instance" {
    when "lisp:lisp-role/lisp:lisp-role-type = 'lisp:etr' or
          lisp:lisp-role/lisp:lisp-role-type = 'lisp:petr'" {
      description
        "Augment is valid when LISP device type is (P)ETR.";
    }
    description
      "This augments LISP devices list with (P)ETR specific
       parameters.";
    container etr {
      presence "LISP (P)ETR operation enabled";
      description
        "(P)ETR parameters.";

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

```
            type inet:ip-address;
            description
              "Map-Server address.";
          }
          leaf auth-key {
            type string;
            description
              "Map-Server authentication key.";
          }
          leaf auth-key-type {
            type lisp:auth-key-type;
            description
              "Map-Server authentication type.";
          }
        }
      }

      container local-eids {
        when "../../lisp:lisp-role/lisp:lisp-role-type = 'lisp:etr'" {
          description
            "Container exists only when LISP device type is ETR.";
        }
        description
          "Virtual networks served by the ETR.";
        list virtual-network {
          key "vni";
          description
            "Virtual network for local-EIDs.";
          leaf vni {
            type 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 "/lisp:lisp/lisp:locator-sets/lisp:locator-set/"
                  + "lisp:locator-set-name";
                }
                description
                  "Locator set mapped to this local EID.";
              }
              leaf record-ttl {
                type uint32;
                units minutes;
                description
                  "Validity period of the EID to RLOCs mapping provided
                  in Map-Replies.";
              }
              leaf want-map-notify {
                type boolean;
                default "true";
                description
                  "Flag which if set in a Map-Register requests that a
                  Map-Notify be sent in response.";
              }
              leaf proxy-reply {
                type boolean;
                default "false";
                description
                  "Flag which if set in a Map-Register requests that the
                  Map-Server proxy Map-Replies for the ETR.";
              }
              leaf registration-interval {
                type uint16;
                units "seconds";
                default "60";
                description
                  "Interval between consecutive Map-Register messages.";
              }
            }
          }
        }
      }
    }
  }
}
<CODE ENDS>
```

5. LISP-Map-Server Module

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

5.1. Module Structure

```
module: ietf-lisp-mapserver
  augment /lisp:lisp/lisp:lisp-router-instances/lisp:lisp-router-instance:
    +--rw map-server!
       +--rw sites
       |  +--rw site* [site-id]
       |     +--rw site-id     uint64
       |     +--rw auth-key
       |        +--rw auth-key-value?   string
       |        +--rw auth-key-type*    lisp:auth-key-type
       +--rw virtual-network-ids
       |  +--rw virtual-network-identifier* [vni]
       |     +--rw vni         lcaf:instance-id-type
       |     +--rw mappings
       |     |  +--rw mapping* [eid-id]
       |     |     +--rw eid-id                        lisp:eid-id
       |     |     +--rw eid-address
       |     |     |  +--rw address-type             lisp-address-family-ref
       |     |     |  +--rw virtual-network-id?      instance-id-type
       |     |     |  +--rw (address)?
       |     |     |     +--:(no-address)
       |     |     |     |  +--rw no-address?              empty
       |     |     |     +--:(ipv4)
       |     |     |     |  +--rw ipv4?                    inet:ipv4-address
       |     |     |     +--:(ipv4-prefix)
       |     |     |     |  +--rw ipv4-prefix?             inet:ipv4-prefix
       |     |     |     +--:(ipv6)
       |     |     |     |  +--rw ipv6?                    inet:ipv6-address
       |     |     |     +--:(ipv6-prefix)
       |     |     |     |  +--rw ipv6-prefix?             inet:ipv6-prefix
       |     |     |     +--:(mac)
       |     |     |     |  +--rw mac?                     yang:mac-address
       |     |     |     +--:(distinguished-name)
       |     |     |     |  +--rw distinguished-name?      distinguished-name-
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
       |     |     +--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
       |     |           +--rw creation-time?      yang:date-and-time
       |     |           +--rw authoritative?      bits
       |     |           +--rw static?             boolean
       |     |           +--rw (locator-list)?
       |     |              +--:(negative-mapping)
       |     |              |  +--rw map-reply-action?   map-reply-action
       |     |              +--:(positive-mapping)
       |     |                 +--rw rlocs
       |     |                    +--rw locator* [id]
       |     |                       +--rw id                    string
       |     |                       +--rw locator-address
       |     |                       |  +--rw address-type             lisp-
address-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
      |     +--ro counters
      |        +--ro map-registers-in?              yang:counter32
      |        +--ro map-registers-in-auth-failed?  yang:counter32
      |        +--ro map-notify-records-out?        yang:counter32
      |        +--ro proxy-reply-records-out?       yang:counter32
      |        +--ro map-requests-forwarded-out?    yang:counter32
      +--rw mapping-system-type?   lisp:mapping-system-ref
      +--ro summary
      |  +--ro number-configured-sites?   uint32
      |  +--ro number-registered-sites?   uint32
      |  +--ro af-datum
      |     +--ro af-data* [address-type]
      |        +--ro address-type             lcaf:lisp-address-family-ref
      |        +--ro number-configured-eids?  uint32
      |        +--ro number-registered-eids?  uint32
      +--ro counters
         +--ro map-registers-in?              yang:counter32
         +--ro map-registers-in-auth-failed?  yang:counter32
         +--ro map-notify-records-out?        yang:counter32
         +--ro proxy-reply-records-out?       yang:counter32
         +--ro map-requests-forwarded-out?    yang:counter32
```

5.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp-mapserver@2018-03-05.yang"
module ietf-lisp-mapserver {
  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-mapserver";
  prefix lisp-ms;
  import ietf-lisp {
    prefix lisp;
  }
  import ietf-lisp-address-types {
    prefix lcaf;
  }
  import ietf-yang-types {
    prefix yang;
    revision-date 2013-07-15;
  }

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

      revision 2018-03-05 {
        description
          "Initial revision.";
        reference
          "https://tools.ietf.org/html/rfc6833";
      }

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

      grouping ms-counters {
          description "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 "/lisp:lisp/lisp:lisp-router-instances"
    + "/lisp:lisp-router-instance" {
      when "lisp:lisp-role/lisp:lisp-role-type = 'lisp-ms:ms'" {
        description
          "Augment is valid when LISP device type is Map-Server.";
      }
      description
        "This augments LISP devices list with Map-Server specific
         parameters.";
      container map-server {
        presence "LISP Map-Server operation enabled";
        description
          "Map-Server parameters.";
          container sites{
            description
              "Sites to accept registrations from.";
```

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

```
                container eid-address {
                  uses lcaf:lisp-address;
                  description
                    "EID in generic LISP address format registered
                     with the Map-Server.";
                }
                leaf-list site-id {
                  type uint64;
                  description "Site ID";
                }
                leaf more-specifics-accepted {
                    type boolean;
                    default "false";
                    description
                      "Flag indicating if more specific prefixes
                       can be registered.";
                }
                leaf mapping-expiration-timeout {
                    type int16;
                    units "seconds";
                    default "180"; //3 times the mapregister int
                    description
                      "Time before mapping is expired if no new
                       registrations are received.";
                }
                container mapping-records {
                  description
                    "Datastore of 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 /lisp:lisp/lisp:lisp-router-instances/lisp:lisp-router-instance:
    +--rw map-resolver!
       +--rw mapping-system-type?   lisp:mapping-system-ref
       +--rw ms-address?            inet:ip-address
```

6.2. Module Definition

```
<CODE BEGINS> file "ietf-lisp-mapresolver@2018-03-05.yang"
module ietf-lisp-mapresolver {
  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-mapresolver";
  prefix lisp-mr;
  import ietf-lisp {
    prefix lisp;
  }
  import ietf-inet-types {
    prefix inet;
  }

  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "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.
    ";
  // RFC Ed.: replace XXXX with actual RFC number and remove
  // this note
  reference "RFC XXXX";

  revision 2018-05-03 {
    description
      "Initial revision.";
    reference
      "https://tools.ietf.org/html/rfc6833";
  }
  identity mr {
    base lisp:lisp-role;
    description
      "LISP Map-Resolver.";
  }
  augment "/lisp:lisp/lisp:lisp-router-instances"
  +"/lisp:lisp-router-instance" {
    when "lisp:lisp-role/lisp:lisp-role-type = 'lisp-mr:mr'" {
      description
        "Augment is valid when LISP device type is Map-Resolver.";
    }
    description
      "This augments LISP devices list with Map-Resolver specific
       parameters.";
    container map-resolver {
      presence "LISP Map-Resolver operation enabled";
      description
        "Map-Resolver parameters.";
      leaf mapping-system-type {
        type lisp:mapping-system-ref;
        description
        "A reference to the mapping system";
      }
      leaf ms-address {
        when "../mapping-system-type='lisp-mr:single-node-mapping-system'";
        type inet:ip-address;
        description
          "address to reach the Map Server when "
```

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

7. LISP-Address-Types Module

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

7.1. Module Definition

```
<CODE BEGINS> file "ietf-lisp-address-types@2018-03-05.yang"
module ietf-lisp-address-types {
  namespace "urn:ietf:params:xml:ns:yang:ietf-lisp-address-types";
  prefix laddr;
  import ietf-inet-types {
    prefix inet;
    revision-date 2013-07-15;
  }
  import ietf-yang-types {
    prefix yang;
    revision-date 2013-07-15;
  }
  organization
    "IETF LISP (Locator/ID Separation Protocol) Working Group";
  contact
    "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 2018-03-05 {
      description
        "Initial revision.";
      reference
        "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10";
    }
    identity lisp-address-family {
      description
        "Base identity from which identities describing LISP address
         families are derived.";
    }
    identity no-address-afi {
      base lisp-address-family;
      description
        "IANA Reserved.";
      }
    identity ipv4-afi {
      base lisp-address-family;
      description
        "IANA IPv4 address family.";
    }
    identity ipv4-prefix-afi {
      base lisp-address-family;
      description
        "IANA IPv4 address family prefix.";
    }
    identity ipv6-afi {
      base lisp-address-family;
      description
        "IANA IPv6 address family.";
    }
    identity ipv6-prefix-afi {
      base lisp-address-family;
```

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

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

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

```
        "Union of address types that can be part of LCAFs.";
    }

    typedef lisp-address-family-ref {
      type identityref {
        base lisp-address-family;
      }
      description
        "LISP address family reference.";
    }
    typedef 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.";
      }
      leaf virtual-network-id {
        type instance-id-type;
        description
          "Virtual Network Identifier (instance-id) of the address.";
      }
      choice address {
        description
          "Various LISP address types, including IP, MAC, and LCAF.";

        leaf no-address {
          when "../address-type = 'laddr:no-addr-afi'" {
            description
              "When AFI is 0.";
          }
          type empty;
          description
            "No address.";
        }
        leaf ipv4 {
          when "../address-type = 'laddr:ipv4-afi'" {
            description
```

```
            "When AFI is IPv4.";
          }
          type inet:ipv4-address;
          description
            "IPv4 address.";
        }
        leaf ipv4-prefix {
          when "../address-type = 'laddr:ipv4-prefix-afi'" {
            description
              "When AFI is IPv4.";
          }
          type inet:ipv4-prefix;
          description
            "IPv4 prefix.";
        }
        leaf ipv6 {
          when "../address-type = 'laddr:ipv6-afi'" {
            description
              "When AFI is IPv6.";
          }
          type inet:ipv6-address;
          description
            "IPv6 address.";
        }
        leaf ipv6-prefix {
          when "../address-type = 'laddr:ipv6-prefix-afi'" {
            description
              "When AFI is IPv6.";
          }
          type inet:ipv6-prefix;
          description
            "IPv6 address.";
        }
        leaf mac {
          when "../address-type = 'laddr:mac-afi'" {
            description
              "When AFI is MAC.";
          }
          type yang:mac-address;
          description
            "MAC address.";
        }
        leaf distinguished-name {
          when "../address-type = 'laddr:distinguished-name-afi'" {
            description
              "When AFI is distinguished-name.";
          }
          type distinguished-name-type;
```

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

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

```
            "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
             #section-4.4";
          leaf address {
              type simple-address;
              description
                "AFI address.";
          }
          leaf protocol {
              type uint8;
              description
                "Protocol number.";
          }
          leaf ip-tos {
              type int32;
              description
                "Type of service field.";
          }
          leaf local-port-low {
              type inet:port-number;
              description
                "Low end of local port range.";
          }
          leaf local-port-high {
              type inet:port-number;
              description
                "High end of local port range.";
          }
          leaf remote-port-low {
              type inet:port-number;
              description
                "Low end of remote port range.";
          }
          leaf remote-port-high {
              type inet:port-number;
              description
                "High end of remote port range.";
          }
        }
        container geo-coordinates {
          when "../address-type = 'laddr:geo-coordinates-lcaf'" {
            description
              "When LCAF type is Geo-coordinates.";
          }
          description
            "Geo-coordinates LCAF type.";
          reference
            "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
             #section-4.5";
```

```
            leaf latitude {
              type bits {
                bit N {
                  description
                    "Latitude bit.";
                }
              }
              description
                "Bit that selects between North and South latitude.";
            }
            leaf latitude-degrees {
              type uint8 {
                range "0 .. 90";
              }
              description
                "Degrees of latitude.";
            }
            leaf latitude-minutes {
              type uint8 {
                range "0..59";
              }
              description
                "Minutes of latitude.";
            }
            leaf latitude-seconds {
              type uint8 {
                range "0..59";
              }
              description
                "Seconds of latitude.";
            }
            leaf longitude {
              type bits {
                bit E {
                  description
                    "Longitude bit.";
                }
              }
              description
                "Bit that selects between East and West longitude.";
            }
            leaf longitude-degrees {
              type uint16 {
                range "0 .. 180";
              }
              description
                "Degrees of longitude.";
            }
```

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

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

```
                description
                  "Lookup bit.";
              }
              bit rloc-probe {
                description
                  "RLOC-probe bit.";
              }
              bit strict {
                description
                  "Strict bit.";
              }
            }
            description
              "Flag bits per hop.";
          }
        }
      }
      container source-dest-key {
        when "../address-type = 'laddr:source-dest-key-lcaf'" {
          description
            "When LCAF type type is Source/Dest.";
        }
        description
          "Source/Dest LCAF type.";
        reference
            "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
            #section-4.11";
        leaf source {
          type simple-address;
          description
            "Source address.";
        }
        leaf dest {
          type simple-address;
          description
            "Destination address.";
        }
      }
      container key-value-address {
        when "../address-type = 'laddr:key-value-address-lcaf'" {
          description
            "When LCAF type type is Key/Value Address.";
        }
        description
          "Key/Value Address LCAF type.";
        reference
          "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
           #section-4.11";
```

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

8. Acknowledgments

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

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

Security Considerations TBD

11. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <https://www.rfc-editor.org/info/rfc6830>.

[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>.

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