LISP Working Group
Internet-Draft
Intended status: Experimental
Expires: September 7, 2019

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

March 6, 2019

LISP YANG Model
draft-ietf-lisp-yang-11

Abstract

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

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 7, 2019.

Copyright Notice

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.

1.1. Requirements Language

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

1.2. Tree Diagrams

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

2. LISP Module

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

2.1. Module Structure

```
module: ietf-lisp
  augment /rt:routing/rt:control-plane-protocols
            /rt:control-plane-protocol:
    +--rw lisp
       +--rw locator-sets
       |  +--rw locator-set* [locator-set-name]
       |     +--rw locator-set-name    string
       |     +--rw (locator-type)?
       |        +--:(local-interface)
       |        |  +--rw interface* [interface-ref]
       |        |     +--rw interface-ref         if:interface-ref
       |        |     +--rw priority?             uint8
       |        |     +--rw weight?               uint8
       |        |     +--rw multicast-priority?   uint8
       |        |     +--rw multicast-weight?     uint8
       |        +--:(general-locator)
       |           +--rw locator* [id]
       |              +--rw id                    string
       |              +--rw locator-address
       |              |  +--rw address-type
       |              |  |       lisp-address-family-ref
       |              |  +--rw (address)?
       |              |     +--:(no-address)
       |              |     |  +--rw no-address?              empty
       |              |     +--:(ipv4)
```

```
|                |      |  +--rw ipv4?
|                |      |           inet:ipv4-address
|                |      +--:(ipv4-prefix)
|                |      |  +--rw ipv4-prefix?
|                |      |           inet:ipv4-prefix
|                |      +--:(ipv6)
|                |      |  +--rw ipv6?
|                |      |           inet:ipv6-address
|                |      +--:(ipv6-prefix)
|                |      |  +--rw ipv6-prefix?
|                |      |           inet:ipv6-prefix
|                |      +--:(mac)
|                |      |  +--rw mac?
|                |      |           yang:mac-address
|                |      +--:(distinguished-name)
|                |      |  +--rw distinguished-name?
|                |      |           distinguished-name-type
|                |      +--:(as-number)
|                |      |  +--rw as-number?
|                |      |           inet:as-number
|                |      +--:(null-address)
|                |      |  +--rw null-address
|                |      |     +--rw address?   empty
|                |      +--:(afi-list)
|                |      |  +--rw afi-list
|                |      |     +--rw address-list*
|                |      |              simple-address
|                |      +--:(instance-id)
|                |      |  +--rw instance-id
|                |      |     +--rw instance-id?
|                |      |     |       instance-id-type
|                |      |     +--rw mask-length?   uint8
|                |      |     +--rw address?       simple-address
|                |      +--:(as-number-lcaf)
|                |      |  +--rw as-number-lcaf
|                |      |     +--rw as?        inet:as-number
|                |      |     +--rw address?   simple-address
|                |      +--:(application-data)
|                |      |  +--rw application-data
|                |      |     +--rw address?
|                |      |     |       simple-address
|                |      |     +--rw protocol?           uint8
|                |      |     +--rw ip-tos?             int32
|                |      |     +--rw local-port-low?
|                |      |     |       inet:port-number
|                |      |     +--rw local-port-high?
|                |      |     |       inet:port-number
|                |      |     +--rw remote-port-low?
```

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

```
          |              +--rw weight?               uint8
          |              +--rw multicast-priority?   uint8
          |              +--rw multicast-weight?     uint8
          +--rw lisp-role* [lisp-role-type]
          |  +--rw lisp-role-type    lisp-role-ref
          +--rw lisp-router-id
          |  +--rw site-id?   uint64
          |  +--rw xtr-id?    lisp:xtr-id-type
          +--rw vpns
             +--rw vpn* [instance-id]
                +--rw instance-id    lcaf:instance-id-type
                +--rw iid-name
                        -> /ni:network-instances/network-instance/name
```

2.2. Module Definition

```
 <CODE BEGINS> file "ietf-lisp@2019-03-05.yang"
 module ietf-lisp {
   yang-version 1.1;

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

   prefix lisp;

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

```
    // RFC Ed.: replace occurences of YYYY with actual RFC number
    // of draft-ietf-rtgwg-ni-model and remove this note
    reference
      "RFC YYYY: YANG Model for Network Instances";
  }

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

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

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

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

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

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

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

  reference "RFC XXXX";

  revision 2019-03-05 {
    description
      "Initial revision.";
    reference
      "https://tools.ietf.org/html/rfc6830";
  }
```

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

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

```
      "Mapping System reference";
  }

  typedef lisp-role-ref {
    type identityref {
      base lisp-role;
    }
    description
      "LISP role reference";
  }
  typedef map-reply-action {
    type enumeration {
      enum no-action {
        value 0;
        description
          "Mapping is kept alive and no encapsulation occurs.";
      }
      enum natively-forward {
        value 1;
        description
          "Matching packets are not encapsulated or dropped but
           natively forwarded.";
      }
      enum send-map-request {
        value 2;
        description
          "Matching packets invoke Map-Requests.";
      }
      enum drop {
        value 3;
        description
          "Matching packets are dropped.";
      }
    }
    description
      "Defines the lisp map-cache ACT type";
    reference "https://tools.ietf.org/html/rfc6830#section-6.1.4";
  }
  typedef eid-id {
    type string;
    description
      "Type encoding of lisp-addresses to be generally used in EID
       keyed lists.";
  }
  typedef auth-algorithm-type {
    type enumeration {
      enum none {
        value 0;
```

```
        description
          "No authentication.";
      }
      enum hmac-sha-1-96 {
        value 1;
        description
          "HMAC-SHA-1-96 (RFC2404) authentication is used.";
      }
      enum hmac-sha-256-128 {
        value 2;
        description
          "HMAC-SHA-256-128 (RFC4868) authentication is used.";
      }
    }
    description
      "Enumeration of the authentication mechanisms supported by
       LISP.";
    reference
      "https://tools.ietf.org/html/rfc6830#section-6.1.6";
  }
  typedef xtr-id-type {
    type binary {
      length "16";
    }
    description
    "128 bit xTR identifier.";
  }

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

```
      description
        "Locator's multicast weight";
    }
  }

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

  }

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

  grouping mapping {
    description
```

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

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

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

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

```
        key "auth-key-id";
        description
        "Authentication key parameters.";
        leaf auth-key-id {
          type string;
          description
          "Identifier of the authentication key.";
        }
        leaf-list auth-algorithm-id {
          type lisp:auth-algorithm-type;
          description
          "Authentication algorithm used with the key.";
        }
        leaf auth-key-value {
          type string;
          description
          "Clear text authentication key.";
        }
      }
    }
  }

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

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

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

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

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

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

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

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

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

3. LISP-ITR Module

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

3.1. Module Structure

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

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

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

```
|      +--:(ipv4)
|      |  +--rw ipv4?
|      |           inet:ipv4-address
|      +--:(ipv4-prefix)
|      |  +--rw ipv4-prefix?
|      |           inet:ipv4-prefix
|      +--:(ipv6)
|      |  +--rw ipv6?
|      |           inet:ipv6-address
|      +--:(ipv6-prefix)
|      |  +--rw ipv6-prefix?
|      |           inet:ipv6-prefix
|      +--:(mac)
|      |  +--rw mac?
|      |           yang:mac-address
|      +--:(distinguished-name)
|      |  +--rw distinguished-name?
|      |           distinguished-name-type
|      +--:(as-number)
|      |  +--rw as-number?
|      |           inet:as-number
|      +--:(null-address)
|      |  +--rw null-address
|      |     +--rw address?   empty
|      +--:(afi-list)
|      |  +--rw afi-list
|      |     +--rw address-list*
|      |              simple-address
|      +--:(instance-id)
|      |  +--rw instance-id
|      |     +--rw instance-id?
|      |     |       instance-id-type
|      |     +--rw mask-length?   uint8
|      |     +--rw address?
|      |              simple-address
|      +--:(as-number-lcaf)
|      |  +--rw as-number-lcaf
|      |     +--rw as?
|      |     |       inet:as-number
|      |     +--rw address?
|      |              simple-address
|      +--:(application-data)
|      |  +--rw application-data
|      |     +--rw address?
|      |     |       simple-address
|      |     +--rw protocol?
|      |     |       uint8
|      |     +--rw ip-tos?
```

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

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

3.2. Module Definition

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

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

  prefix lisp-itr;

  // RFC Ed.: replace occurences of XXXX with actual RFC number
  // and remove this note
  import ietf-lisp {
    prefix lisp;
    reference "RFC XXXX: LISP YANG model";
  }
  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }
```

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

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

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

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

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

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

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

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

    reference "RFC XXXX";

    revision 2019-02-23 {
      description
        "Initial revision.";
      reference
        "https://tools.ietf.org/html/rfc6830";
    }
```

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

```
        }
        container map-resolvers {
          description
            "Map-Resolvers that the (P)ITR uses.";
          leaf-list map-resolver {
            type inet:ip-address;
            description
              "Each Map-Resolver within the list of Map-Resolvers.";
          }
        }
        container proxy-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 {
          leaf size {
            type uint32;
            config false;
            description
              "Current number of entries in the EID-to-RLOC map-cache";
          }
          leaf limit {
            type uint32;
            config false;
            description
              "Maximum permissible number of entries in the EID-to-RLOC
               map-cache";
          }

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

4. LISP-ETR Module

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

4.1. Module Structure

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

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

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

4.2. Module Definition

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

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

  prefix lisp-etr;

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

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

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

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

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

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

     Redistribution and use in source and binary forms, with or
```

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

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

  reference "RFC XXXX";

  revision 2019-02-23 {
    description
      "Initial revision.";
    reference
      "https://tools.ietf.org/html/rfc6830";
  }
  augment "/rt:routing/rt:control-plane-protocols"
        + "/rt:control-plane-protocol/lisp:lisp" {
    when "lisp:lisp-role/lisp:lisp-role-type = 'lisp:etr' or
          lisp:lisp-role/lisp:lisp-role-type = 'lisp:petr'" {
      description
        "Augment is valid when LISP device type is (P)ETR.";
    }
    description
      "This augments LISP devices list with (P)ETR specific
       parameters.";
    container etr {
      presence "LISP (P)ETR operation enabled";
      description
        "(P)ETR parameters.";

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

```
          }
          uses lisp:auth-key;
        }
      }

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

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

5. LISP-Map-Server Module

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

5.1. Module Structure

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

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

```
|       |       |       |      +--rw address?
|       |       |       |             simple-address
|       |       |       +--:(nat-traversal)
|       |       |       |  +--rw nat-traversal
|       |       |       |     +--rw ms-udp-port?        uint16
|       |       |       |     +--rw etr-udp-port?       uint16
|       |       |       |     +--rw global-etr-rloc?
|       |       |       |     |       simple-address
|       |       |       |     +--rw ms-rloc?
|       |       |       |     |       simple-address
|       |       |       |     +--rw private-etr-rloc?
|       |       |       |     |       simple-address
|       |       |       |     +--rw rtr-rlocs*
|       |       |       |             simple-address
|       |       |       +--:(explicit-locator-path)
|       |       |       |  +--rw explicit-locator-path
|       |       |       |     +--rw hop* [hop-id]
|       |       |       |        +--rw hop-id      string
|       |       |       |        +--rw address?    simple-address
|       |       |       |        +--rw lrs-bits?   bits
|       |       |       +--:(source-dest-key)
|       |       |       |  +--rw source-dest-key
|       |       |       |     +--rw source?   simple-address
|       |       |       |     +--rw dest?     simple-address
|       |       |       +--:(key-value-address)
|       |       |       |  +--rw key-value-address
|       |       |       |     +--rw key?     simple-address
|       |       |       |     +--rw value?   simple-address
|       |       |       +--:(service-path)
|       |       |          +--rw service-path
|       |       |             +--rw service-path-id?
|       |       |             |       service-path-id-type
|       |       |             +--rw service-index?     uint8
|       |       +--rw site-id*                        uint64
|       |       +--rw more-specifics-accepted?       boolean
|       |       +--rw mapping-expiration-timeout?    int16
|       |       +--ro first-registration-time?
|       |       |       yang:date-and-time
|       |       +--ro last-registration-time?
|       |       |       yang:date-and-time
|       |       +--rw mapping-records
|       |          +--rw mapping-record* [xtr-id]
|       |             +--rw xtr-id
|       |             |       lisp:xtr-id-type
|       |             +--rw site-id?                  uint64
|       |             +--rw eid
|       |             |  +--rw address-type
|       |             |  |       lisp-address-family-ref
```

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

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

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

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

```
|       |                                 |       |       |       uint8
|       |                                 |       |       +--rw longitude?
|       |                                 |       |       |       bits
|       |                                 |       |       +--rw longitude-degrees?
|       |                                 |       |       |       uint16
|       |                                 |       |       +--rw longitude-minutes?
|       |                                 |       |       |       uint8
|       |                                 |       |       +--rw longitude-seconds?
|       |                                 |       |       |       uint8
|       |                                 |       |       +--rw altitude?
|       |                                 |       |       |       int32
|       |                                 |       |       +--rw address?
|       |                                 |       |               simple-address
|       |                                 |       +--:(nat-traversal)
|       |                                 |       |  +--rw nat-traversal
|       |                                 |       |       +--rw ms-udp-port?
|       |                                 |       |       |       uint16
|       |                                 |       |       +--rw etr-udp-port?
|       |                                 |       |       |       uint16
|       |                                 |       |       +--rw global-etr-rloc?
|       |                                 |       |       |       simple-address
|       |                                 |       |       +--rw ms-rloc?
|       |                                 |       |       |       simple-address
|       |                                 |       |       +--rw private-etr-rloc?
|       |                                 |       |       |       simple-address
|       |                                 |       |       +--rw rtr-rlocs*
|       |                                 |       |               simple-address
|       |                                 |       +--:(explicit-locator-path)
|       |                                 |       |  +--rw explicit-locator-path
|       |                                 |       |       +--rw hop* [hop-id]
|       |                                 |       |          +--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:counter64
      |         +--ro map-registers-in-auth-failed?   yang:counter64
      |         +--ro map-notify-records-out?         yang:counter64
      |         +--ro proxy-reply-records-out?        yang:counter64
      |         +--ro map-requests-forwarded-out?     yang:counter64
      +--rw mapping-system-type?   lisp:mapping-system-ref
      +--ro summary
      |  +--ro number-configured-sites?   uint32
      |  +--ro number-registered-sites?   uint32
      |  +--ro af-datum
      |     +--ro af-data* [address-type]
      |        +--ro address-type
      |        |       lcaf:lisp-address-family-ref
      |        +--ro number-configured-eids?   uint32
      |        +--ro number-registered-eids?   uint32
      +--ro counters
         +--ro map-registers-in?               yang:counter64
         +--ro map-registers-in-auth-failed?   yang:counter64
         +--ro map-notify-records-out?         yang:counter64
         +--ro proxy-reply-records-out?        yang:counter64
         +--ro map-requests-forwarded-out?     yang:counter64
```

5.2. Module Definition

```
  <CODE BEGINS> file "ietf-lisp-mapserver@2019-03-05.yang"
  module ietf-lisp-mapserver {
    yang-version 1.1;

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

    prefix lisp-ms;

    // RFC Ed.: replace occurences of XXXX with actual RFC number
```

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

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

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

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

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

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

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

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

    reference "RFC XXXX";

    revision 2019-03-05 {
      description
        "Initial revision.";
      reference
        "https://tools.ietf.org/html/rfc6833";
    }

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

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

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

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

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

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

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

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

```
            key "instance-id";
            description
              "VPN instances in the Map-Server.";
            leaf instance-id {
              type lcaf:instance-id-type;
              description
                "VPN identifier.";
            }
            container mappings {
              description
                "EIDs registered by device.";
              list mapping {
                key "eid-id";
                description
                  "List of EIDs registered by device.";
                leaf eid-id {
                  type lisp:eid-id;
                  description
                    "Id of the EID registered.";
                }
                container eid-address {
                  uses lcaf:lisp-address;
                  description
                    "EID in generic LISP address format registered
                     with the Map-Server.";
                }
                leaf-list site-id {
                  type uint64;
                  description "Site ID";
                }
                leaf more-specifics-accepted {
                    type boolean;
                    default "false";
                    description
                      "Flag indicating if more specific prefixes
                       can be registered.";
                }
                leaf mapping-expiration-timeout {
                    type int16;
                    units "seconds";
                    default "180"; //3 times the mapregister int
                    description
                      "Time before mapping is expired if no new
                       registrations are received.";
                }
                leaf first-registration-time {
                    type yang:date-and-time;
                    config false;
```

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

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

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

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

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

6. LISP-Map-Resolver Module

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

6.1. Module Structure

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

6.2. Module Definition

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

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

  prefix lisp-mr;

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

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

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

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

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

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

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

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

  reference "RFC XXXX";

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

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

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

7. LISP-Address-Types Module

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

7.1. Module Definition

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

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

  prefix laddr;

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

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

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

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

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

```
        define vendor-specific parameters.

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

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

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

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

    revision 2019-02-23 {
      description
        "Initial revision.";
      reference
        "RC8060: LISP Canonical Address Format (LCAF)";
    }
    identity lisp-address-family {
      description
        "Base identity from which identities describing LISP address
         families are derived.";
    }
    identity no-address-afi {
      base lisp-address-family;
      description
        "IANA Reserved.";
    }
    identity ipv4-afi {
      base lisp-address-family;
      description
        "IANA IPv4 address family.";
    }
    identity ipv4-prefix-afi {
      base lisp-address-family;
      description
        "IANA IPv4 address family prefix.";
    }
    identity ipv6-afi {
      base lisp-address-family;
```

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

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

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

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

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

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

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

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

```
          description
            "Distinguished Name address.";
        }
        leaf as-number {
          when "../address-type = 'laddr:as-number-afi'" {
            description
              "When AFI is as-number.";
          }
          type inet:as-number;
          description
            "AS Number.";
        }
        container null-address {
          when "../address-type = 'laddr:null-address-lcaf'" {
            description
              "When LCAF type is null.";
          }
          description
            "Null body LCAF type";
          leaf address {
            type empty;
            description
              "AFI address.";
          }
        }
        container afi-list {
          when "../address-type = 'laddr:afi-list-lcaf'" {
            description
              "When LCAF type is AFI-List.";
          }
          description
            "AFI-List LCAF type.";
          reference
            "http://tools.ietf.org/html/draft-ietf-lisp-lcaf-10
             #section-4.16.1";
          leaf-list address-list {
            type simple-address;
            description
              "List of AFI addresses.";
          }
        }
        container instance-id {
          when "../address-type = 'laddr:instance-id-lcaf'" {
            description
              "When LCAF type is Instance ID as per RFC8060.";
          }
          description
            "Instance ID LCAF type.";
```

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

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

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

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

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

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

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

7.2. Data Model examples

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

7.2.1. LISP protocol instance

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

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

7.2.2. LISP ITR

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

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

7.2.3. LISP ETR

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

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

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

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

7.2.4. LISP Map-Server

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

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

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

8. Acknowledgments

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

9. IANA Considerations

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

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

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

Registrant Contact: The IESG.

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

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

Registrant Contact: The IESG.

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

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

Registrant Contact: The IESG.

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

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

Registrant Contact: The IESG.

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

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

Registrant Contact: The IESG.

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

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

Registrant Contact: The IESG.

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

---

10. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a pre-configured subset of all available NETCONF or RESTCONF protocol operations and content.

The security considerations of LISP control-plane [RFC6833] and LISP data-plane [RFC6830] as well as the LISP threat analysis [RFC7835] apply to this YANG model.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
lisp:lisp/

Access to the locator-sets node may modify which interfaces are used for data and/or control traffic as well as affect the load balancing of data-plane traffic. Access to the lisp-role node may prevent the device from perform its intended data-plane and/or control-plane operation. Access to the router-id node allows to modify the unique identifier of the device, which may result in disruption of its LISP control-plane operation. Access to the vpn node may allow to redirect data-plane traffic to erroneous local or remote network instances.

/rt:routing/rt:control-plane-protocols/rt:control-plane-
protocol/lisp:lisp/lisp:map-server

Access to the sites node can prevent authorized devices from registering mappings in the Map-Server and/or allow unauthorized devices to so. Access to the vpn node can result in corrupted mapping sate that may propagate across the LISP network, potentially resulting in forwarding of data-plane traffic to arbitrary destinations and general disruption of the data-plane operation. Access to mapping-system-type and/or ddt-mapping-system nodes may prevent the device to connect to the Mapping System infrastructure and consequentially to attract Map-Request messages.

/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp/lisp:map-resolver

Access to mapping-system-type, ms-address and/or ddt-mapping-system nodes may prevent the device to connect to the Mapping System infrastructure and forward Map-Request messages.

/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp/lisp:itr

Access to the rloc-probing node can increase the control-plane overhead in the device or affect the capability of the device to detect failures on the underlay. Access to the itr-rlocs node may prevent the device from getting Map-Reply messages. Access to the map-resolvers node can prevent the device from sending its Map-Request messages to valid Map-Resolvers. Access to the proxy-etrs nodes can affect the capability of the device to send data-plane traffic towards non-LISP destinations. Access to the map-cache node can result in forwarding of data-plane traffic to arbitrary destinations and general disruption of data-plane operation.

/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp/lisp:etr

Access to the map-servers node can prevent the device from registering its local mappings into the Mapping System. Access to the local-eids node can disrupt data-plane operation on the device and/or result in the device registering corrupted mappings into the Mapping System.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp

Access to the locator-sets node can expose the locators the device is using for its control and/or data operation. Access to the lisp-role node can disclose the LISP roles instantiated at the device which facilitates mounting attacks against the device. Access to the router-id node can expose the unique identifier of device which may allow a third party to track its control-plane operation and/or impersonate the device. Access to the vpn node can leak the local mapping between LISP Instance IDs and local network instances.

/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp/lisp:map-server

Access to the sites node can expose the credentials used to register mappings and allow unauthorized devices to do so. Access to the vpn node can expose the mappings currently registered in the device, which has privacy implications. Access to the mapping-system-type node may reveal the Mapping System in use which can be used to mount attacks against the device and/or the Mapping System. Access to the summary and counters nodes may expose operational statistics of the device.

/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp/lisp:map-resolver

Access to the mapping-system-type node may reveal the Mapping System in use which can be used to mount attacks against the device and/or the Mapping System. Access to the ms-address and/or ddt-mapping-system nodes can leak the information about the Mapping System infrastructure used by the device, which can be used to block communication and/or mount attacks against it.

/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp/lisp:itr

Access to the rloc-probing node can expose if and how the device is using control-plane signaling to probe underlay locators. Access to the itr-rlocs node may disclose the addresses the device is using to receive Map-Reply messages. Access to the map-resolvers node can expose the Map-Resolvers used by the device, which can be used to mount attacks against the device and/or the Mapping System. Access to the proxy-etrs node can disclose the PETRs used by the device, which can be used to mount attacks against the device and/or PETRs. Access to the map-cache node can expose the mappings currently cached in the device, which has privacy implications.

/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/lisp:lisp/lisp:etr

Access to the map-servers node can expose the credentials used by the device to register mappings into the Mapping System allowing an unauthorized device to impersonate and register mappings on behalf the authorized device. Access to the local-eids node can expose the local EIDs currently being served by the device, which has privacy implications.

11. Normative References

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

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

[RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <https://www.rfc-editor.org/info/rfc6832>.

[RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, DOI 10.17487/RFC6833, January 2013, <https://www.rfc-editor.org/info/rfc6833>.

[RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, DOI 10.17487/RFC6836, January 2013, <https://www.rfc-editor.org/info/rfc6836>.

[RFC7835] Saucez, D., Iannone, L., and O. Bonaventure, "Locator/ID Separation Protocol (LISP) Threat Analysis", RFC 7835, DOI 10.17487/RFC7835, April 2016, <https://www.rfc-editor.org/info/rfc7835>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

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

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Authors' Addresses

Vina Ermagan
Google
USA

Email: ermagan@gmail.com

Alberto Rodriguez-Natal
Cisco Systems
San Jose, CA
USA

Email: natal@cisco.com

Florin Coras
Cisco Systems
San Jose, CA
USA

Email: fcoras@cisco.com

Carl Moberg
Cisco Systems
San Jose, CA
USA

Email: camoberg@cisco.com

Reshad Rahman
Cisco Systems
Canada

Email: rrahman@cisco.com

Albert Cabellos-Aparicio
Technical University of Catalonia
Barcelona
Spain

Email: acabello@ac.upc.edu

Fabio Maino
Cisco Systems
San Jose, CA
USA

Email: fmaino@cisco.com