PIM Working Group
Internet-Draft
Intended Status: Standard Track
Expires: April 19, 2019

X. Liu
Volta
F. Guo
Huawei
M. Sivakumar
Juniper
P. McAllister
Metaswitch Networks
A. Peter
Individual

October 19, 2018

A YANG data model for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD)
draft-ietf-pim-igmp-mld-yang-09

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on April 15, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

This document defines a YANG data model that can be used to configure and manage Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) devices.

Table of Contents

1. Introduction 2
   1.1. Terminology 3
   1.2. Tree Diagrams 3
   1.3. Prefixes in Data Node Names..... 3
2. Design of Data model..... 4
   2.1. Scope of model 4
   2.2. Optional capabilities..... 4
   2.3. Position of address family in hierarchy..... 5
3. Module Structure 5
   3.1. IGMP Configuration and Operational state..... 5
   3.2. MLD Configuration and Operational State..... 7
   3.3. IGMP and MLD RPC..... 9
4. IGMP and MLD YANG Modules..... 10
5. Security Considerations..... 33
6. IANA Considerations 35
7. Acknowledgments 35
8. Contributing Authors..... 36
9. References 36
   9.1. Normative References..... 36
   9.2. Informative References..... 38

1. Introduction

YANG [RFC6020] [RFC7950] is a data definition language that was introduced to model the configuration and running state of a device managed using network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. YANG is now also being used as a component of wider management interfaces, such as CLIs.

This document defines a YANG data model that can be used to configure and manage Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) devices. This model will support the core IGMP and MLD protocols, as well as many other features

mentioned in separate IGMP and MLD RFCs. Non-core features are defined as optional in the provided data model.

1.1. Terminology

The terminology for describing YANG data models is found in [RFC6020] [RFC7950].

The following abbreviations are used in this document and the defined model:

IGMP:

Internet Group Management Protocol [RFC3376].

MLD:

Multicast Listener Discovery [RFC3810].

1.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

1.3. Prefixes in Data Node Names

In this document, names of data nodes, actions, and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the

Prefix	YANG module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
if	ietf-interfaces	[RFC8343]
ip	ietf-ip	[RFC8344]
rt	ietf-routing	[RFC8349]
rt-types	ietf-routing-types	[RFC8294]

```
| acl       | ietf-access-control-list | [I-D.ietf-acl-yang] |

+-----------+--------------------------+---------------------+
```

Table 1: Prefixes and Corresponding YANG Modules

2. Design of Data model

2.1. Scope of model

The model covers IGMPv1 [RFC1112], IGMPv2 [RFC2236], IGMPv3 [RFC3376] and MLDv1 [RFC2710], MLDv2 [RFC3810].

The configuration of IGMP and MLD features, and the operational state fields and RPC definitions are not all included in this document of the data model. This model can be extended, though the structure of what has been written may be taken as representative of the structure of the whole model.

This model does not cover other IGMP and MLD related protocols such as IGMP/MLD Proxy [RFC4605] or IGMP/MLD Snooping [RFC4541] etc., these will be specified in separate documents.

2.2. Optional capabilities

This model is designed to represent the capabilities of IGMP and MLD devices with various specifications, including some with basic subsets of the IGMP and MLD protocols. The main design goals of this document are that any major now-existing implementation may be said to support the basic model, and that the configuration of all implementations meeting the specification is easy to express through some combination of the features in the basic model and simple vendor augmentations.

There is also value in widely-supported features being standardized, to save work for individual vendors, and so that mapping between different vendors' configuration is not needlessly complicated. Therefore these modules declare a number of features representing capabilities that not all deployed devices support.

The extensive use of feature declarations should also substantially simplify the capability negotiation process for a vendor's IGMP and MLD implementations.

On the other hand, operational state parameters are not so widely designated as features, as there are many cases where the defaulting of an operational state parameter would not cause any harm to the

system, and it is much more likely that an implementation without native support for a piece of operational state would be able to derive a suitable value for a state variable that is not natively supported.

2.3. Position of address family in hierarchy

The current document contains IGMP and MLD as separate schema branches in the structure. The reason for this is to make it easier for implementations which may optionally choose to support specific address families. And the names of objects may be different between the IPv4 (IGMP) and IPv6 (MLD) address families.

3. Module Structure

3.1. IGMP Configuration and Operational state

The IGMP YANG model conforms to the Network Management Datastore Architecture (NMDA) [RFC8342]. The operational state data is combined with the associated configuration data in the same hierarchy [I-D.ietf-netmod-rfc6087bis]. The IGMP module defines in a three-level hierarchy structure as listed below:

Global level: IGMP configuration and operational state attributes for the entire routing system.

Interface-global: Only including configuration data nodes that IGMP configuration attributes are applicable to all the interfaces whose interface-level corresponding attributes are not existing, with same attributes' value for these interfaces.

Interface-level: IGMP configuration and operational state attributes specific to the given interface.

Where fields are not genuinely essential to protocol operation, they are marked as optional. Some fields will be essential but have a default specified, so that they need not be configured explicitly.

This model augments the core routing data model "ietf-routing" specified in [RFC8349]. The IGMP model augments "/rt:routing/rt:control-plane-protocols" as opposed to augmenting "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol", as the latter would allow multiple protocol instances, while the IGMP protocol is designed to be enabled or disabled as a single protocol instance on a network instance or a logical network element.

```
  augment /rt:routing/rt:control-plane-protocols:
    +--rw igmp {feature-igmp}?
       +--rw global
```

```
          |  +--rw enable?          boolean {global-admin-enable}?
          |  +--rw max-entries?     uint32 {global-max-entries}?
          |  +--rw max-groups?      uint32 {global-max-groups}?
          |  +--ro entries-count?   uint32
          |  +--ro groups-count?    uint32
          |  +--ro statistics
          |     +--ro discontinuity-time?   yang:date-and-time
          |     +--ro error
          |     |  +--ro total?       yang:counter64
          |     |  +--ro query?       yang:counter64
          |     |  +--ro report?      yang:counter64
          |     |  +--ro leave?       yang:counter64
          |     |  +--ro checksum?    yang:counter64
          |     |  +--ro too-short?   yang:counter64
          |     +--ro received
          |     |  +--ro total?    yang:counter64
          |     |  +--ro query?    yang:counter64
          |     |  +--ro report?   yang:counter64
          |     |  +--ro leave?    yang:counter64
          |     +--ro sent
          |        +--ro total?    yang:counter64
          |        +--ro query?    yang:counter64
          |        +--ro report?   yang:counter64
          |        +--ro leave?    yang:counter64
          +--rw interfaces
             +--rw last-member-query-interval?   uint16
             +--rw query-interval?               uint16
             +--rw query-max-response-time?      uint16
             +--rw require-router-alert?         boolean {intf-require-router-
alert}?
             +--rw robustness-variable?          uint8
             +--rw version?                      uint8
             +--rw max-groups-per-interface?     uint32 {intf-max-groups}?
             +--rw interface* [interface-name]
                +--rw interface-name                if:interface-ref
                +--rw last-member-query-interval?   uint16
                +--rw query-interval?               uint16
                +--rw query-max-response-time?      uint16
                +--rw require-router-alert?         boolean {intf-require-
router-alert}?
                +--rw robustness-variable?          uint8
                +--rw version?                      uint8
                +--rw enable?                       boolean {intf-admin-
enable}?
                +--rw group-policy?                 -> /acl:acls/acl/name
                +--rw immediate-leave?              empty {intf-immediate-
leave}?
                +--rw max-groups?                   uint32 {intf-max-groups}?
```

```
                +--rw max-group-sources?            uint32 {intf-max-group-
sources}?
                +--rw source-policy?                -> /acl:acls/acl/name
{intf-source-policy}?
                +--rw verify-source-subnet?         empty {intf-verify-source-
subnet}?
                +--rw explicit-tracking?            empty {intf-explicit-
tracking}?
                +--rw exclude-lite?                 empty {intf-exclude-lite}?
```

```
                +--rw join-group*                   rt-types:ipv4-multicast-
group-address {intf-join-group}?
                +--rw ssm-map* [ssm-map-source-addr ssm-map-group-policy]
{intf-ssm-map}?
                |  +--rw ssm-map-source-addr     ssm-map-ipv4-addr-type
                |  +--rw ssm-map-group-policy    string
                +--rw static-group* [group-addr source-addr] {intf-static-
group}?
                |  +--rw group-addr     rt-types:ipv4-multicast-group-address
                |  +--rw source-addr    rt-types:ipv4-multicast-source-address
                +--ro oper-status                   enumeration
                +--ro querier                       inet:ipv4-address
                +--ro joined-group*                 rt-types:ipv4-multicast-
group-address {intf-join-group}?
                +--ro group* [group-address]
                   +--ro group-address    rt-types:ipv4-multicast-group-address
                   +--ro expire           uint32
                   +--ro filter-mode      enumeration
                   +--ro up-time          uint32
                   +--ro last-reporter?   inet:ipv4-address
                   +--ro source* [source-address]
                      +--ro source-address    inet:ipv4-address
                      +--ro expire            uint32
                      +--ro up-time           uint32
                      +--ro host-count?       uint32 {intf-explicit-tracking}?
                      +--ro last-reporter?    inet:ipv4-address
                      +--ro host* [host-address] {intf-explicit-tracking}?
                         +--ro host-address        inet:ipv4-address
                         +--ro host-filter-mode    enumeration
```

3.2. MLD Configuration and Operational State

The MLD YANG model uses the same structure as IGMP YANG model. The MLD module also defines in a three-level hierarchy structure as listed below:

```
  augment /rt:routing/rt:control-plane-protocols:
    +--rw mld {feature-mld}?
       +--rw global
       |  +--rw enable?          boolean {global-admin-enable}?
       |  +--rw max-entries?     uint32 {global-max-entries}?
       |  +--rw max-groups?      uint32 {global-max-groups}?
       |  +--ro entries-count?   uint32
       |  +--ro groups-count?    uint32
       |  +--ro statistics
       |     +--ro discontinuity-time?   yang:date-and-time
       |     +--ro error
```

```
          |     |  +--ro total?       yang:counter64
          |     |  +--ro query?       yang:counter64
          |     |  +--ro report?      yang:counter64
          |     |  +--ro leave?       yang:counter64
          |     |  +--ro checksum?    yang:counter64
```

```
          |     |  +--ro too-short?   yang:counter64
          |     +--ro received
          |     |  +--ro total?    yang:counter64
          |     |  +--ro query?    yang:counter64
          |     |  +--ro report?   yang:counter64
          |     |  +--ro leave?    yang:counter64
          |     +--ro sent
          |        +--ro total?    yang:counter64
          |        +--ro query?    yang:counter64
          |        +--ro report?   yang:counter64
          |        +--ro leave?    yang:counter64
          +--rw interfaces
             +--rw last-member-query-interval?   uint16
             +--rw query-interval?               uint16
             +--rw query-max-response-time?      uint16
             +--rw require-router-alert?         boolean {intf-require-router-
alert}?
             +--rw robustness-variable?          uint8
             +--rw version?                      uint8
             +--rw max-groups-per-interface?     uint32 {intf-max-groups}?
             +--rw interface* [interface-name]
                +--rw interface-name                if:interface-ref
                +--rw last-member-query-interval?   uint16
                +--rw query-interval?               uint16
                +--rw query-max-response-time?      uint16
                +--rw require-router-alert?         boolean {intf-require-
router-alert}?
                +--rw robustness-variable?          uint8
                +--rw version?                      uint8
                +--rw enable?                       boolean {intf-admin-
enable}?
                +--rw group-policy?                 -> /acl:acls/acl/name
                +--rw immediate-leave?              empty {intf-immediate-
leave}?
                +--rw max-groups?                   uint32 {intf-max-groups}?
                +--rw max-group-sources?            uint32 {intf-max-group-
sources}?
                +--rw source-policy?                -> /acl:acls/acl/name
{intf-source-policy}?
                +--rw verify-source-subnet?         empty {intf-verify-source-
subnet}?
                +--rw explicit-tracking?            empty {intf-explicit-
tracking}?
                +--rw exclude-lite?                 empty {intf-exclude-lite}?
                +--rw join-group*                   rt-types:ipv6-multicast-
group-address {intf-join-group}?
                +--rw ssm-map* [ssm-map-source-addr ssm-map-group-policy]
{intf-ssm-map}?
```

```
                |  +--rw ssm-map-source-addr     ssm-map-ipv6-addr-type
                |  +--rw ssm-map-group-policy    string
                +--rw static-group* [group-addr source-addr] {intf-static-
group}?
                |  +--rw group-addr     rt-types:ipv6-multicast-group-address
                |  +--rw source-addr    rt-types:ipv6-multicast-source-address
                +--ro oper-status                   enumeration
                +--ro querier                       inet:ipv6-address
                +--ro joined-group*                 rt-types:ipv6-multicast-
group-address {intf-join-group}?
                +--ro group* [group-address]
                   +--ro group-address    rt-types:ipv6-multicast-group-address
                   +--ro expire           uint32
```

```
                  +--ro filter-mode      enumeration
                  +--ro up-time          uint32
                  +--ro last-reporter?   inet:ipv6-address
                  +--ro source* [source-address]
                     +--ro source-address    inet:ipv6-address
                     +--ro expire            uint32
                     +--ro up-time           uint32
                     +--ro host-count?       uint32 {intf-explicit-tracking}?
                     +--ro last-reporter?    inet:ipv6-address
                     +--ro host* [host-address] {intf-explicit-tracking}?
                        +--ro host-address        inet:ipv6-address
                        +--ro host-filter-mode    enumeration
```

3.3. IGMP and MLD RPC

IGMP and MLD RPC clears the specified IGMP and MLD group membership.

```
  rpcs:

    +---x clear-igmp-groups {rpc-clear-groups}?

    |  +---w input

    |     +---w interface-name?   -> /rt:routing/control-plane-
protocols/igmp-mld:igmp/interfaces/interface/interface-name
{feature-igmp}?

    |     +---w group-address?    rt-types:ipv4-multicast-group-
address

    |     +---w source-address?   rt-types:ipv4-multicast-source-
address

    +---x clear-mld-groups {rpc-clear-groups}?

       +---w input

          +---w interface-name?   -> /rt:routing/control-plane-
protocols/igmp-mld:mld/interfaces/interface/interface-name {feature-
mld}?

          +---w group-addrss?     rt-types:ipv6-multicast-group-
address

          +---w source-address?   rt-types:ipv6-multicast-source-
address
```

4. IGMP and MLD YANG Modules

```
<CODE BEGINS> file "ietf-igmp-mld@2018-09-15.yang"
module ietf-igmp-mld {
   yang-version 1.1;
   namespace "urn:ietf:params:xml:ns:yang:ietf-igmp-mld";
   // replace with IANA namespace when assigned
   prefix igmp-mld;

   import ietf-inet-types {
     prefix "inet";
   }

   import ietf-yang-types {
     prefix "yang";
   }

   import ietf-routing-types {
     prefix "rt-types";
   }

   import ietf-access-control-list {
     prefix "acl";
   }

   import ietf-routing {
     prefix "rt";
   }

   import ietf-interfaces {
     prefix "if";
   }

   import ietf-ip {
     prefix ip;
   }

   organization
     "IETF PIM Working Group";

   contact
     "WG Web:   <http://tools.ietf.org/wg/pim/>
      WG List:  <mailto:pim@ietf.org>

      WG Chair: Stig Venaas
                <mailto:stig@venaas.com>

      WG Chair: Mike McBride
                <mailto:mmcbride7@gmail.com>
```

```
         Editor:   Xufeng Liu
                   <mailto:xufeng.liu.ietf@gmail.com>

         Editor:   Feng Guo
                   <mailto:guofeng@huawei.com>

         Editor:   Mahesh Sivakumar
                   <mailto:sivakumar.mahesh@gmail.com>

         Editor:   Pete McAllister
                   <mailto:pete.mcallister@metaswitch.com>

         Editor:   Anish Peter
                   <mailto:anish.ietf@gmail.com>";

      description
        "The module defines a collection of YANG definitions common for
        IGMP and MLD.";

      revision 2018-09-15 {
        description
          "Updated yang data model for default value, address type and
           repeated leaf definition.";
        reference
          "RFC XXXX: A YANG Data Model for IGMP and MLD";
      }
      revision 2018-06-21 {
        description
          "Updated yang data model for parameter range and description.";
        reference
          "RFC XXXX: A YANG Data Model for IGMP and MLD";
      }
      revision 2017-10-20 {
        description
          "Updated yang data model for adding explicit-tracking and
           lightweight IGMPv3 and MLDv2 function.";
        reference
          "RFC XXXX: A YANG Data Model for IGMP and MLD";
      }
      revision 2017-09-19 {
        description
          "Updated yang data model for NMDA version and errata.";
        reference
          "RFC XXXX: A YANG Data Model for IGMP and MLD";
      }

      /*
       * Features
```

```
 */
feature feature-igmp {
  description
    "Support IGMP protocol for IPv4 group membership record.";
}
feature feature-mld {
  description
    "Support MLD protocol for IPv6 group membership record.";
}
feature global-admin-enable {
  description
    "Support global configuration to enable or disable protocol.";
}

feature global-interface-config {
  description
    "Support global configuration applied for all interfaces.";
}

feature global-max-entries {
  description
    "Support configuration of global max-entries.";
}

feature global-max-groups {
  description
    "Support configuration of global max-groups.";
}

feature intf-admin-enable {
  description
    "Support configuration of interface administrative enabling.";
}

feature intf-immediate-leave {
  description
    "Support configuration of interface immediate-leave.";
}

feature intf-join-group {
  description
    "Support configuration of interface join-group.";
}

feature intf-max-groups {
  description
    "Support configuration of interface max-groups.";
}
```

```
      feature intf-max-group-sources {
        description
          "Support configuration of interface max-group-sources.";
      }

      feature intf-require-router-alert {
        description
          "Support configuration of interface require-router-alert.";
      }

      feature intf-source-policy {
        description
          "Support configuration of interface source policy.";
      }

      feature intf-ssm-map {
        description
          "Support configuration of interface ssm-map.";
      }

      feature intf-static-group {
        description
          "Support configuration of interface static-group.";
      }

      feature intf-verify-source-subnet {
        description
          "Support configuration of interface verify-source-subnet.";
      }

      feature intf-explicit-tracking {
        description
          "Support configuration of interface explicit-tracking hosts.";
      }

      feature intf-exclude-lite {
        description
          "Support configuration of interface exclude-lite.";
      }

      feature per-interface-config {
        description
          "Support per interface configuration.";
      }

      feature rpc-clear-groups {
        description
          "Support rpc's to clear groups.";
      }
```

```
/*
 * Typedefs
 */
typedef ssm-map-ipv4-addr-type {
  type union {
    type enumeration {
      enum 'policy' {
        description
        "Source address is specified in SSM map policy.";
      }
    }
    type inet:ipv4-address;
  }
  description
    "Multicast source IP address type for SSM map.";
} // source-ipv4-addr-type

typedef ssm-map-ipv6-addr-type {
  type union {
    type enumeration {
      enum 'policy' {
        description
        "Source address is specified in SSM map policy.";
      }
    }
    type inet:ipv6-address;
  }
  description
    "Multicast source IP address type for SSM map.";
} // source-ipv6-addr-type

/*
 * Identities
 */

/*
 * Groupings
 */
grouping global-config-attributes {
  description "Global IGMP and MLD configuration.";

  leaf enable {
    if-feature global-admin-enable;
    type boolean;
    default false;
    description
      "true to enable IGMP or MLD in the routing instance;
```

```
            false to disable IGMP or MLD in the routing instance.";
```

```
        }

        leaf max-entries {
          if-feature global-max-entries;
          type uint32;
          description
            "The maximum number of entries in IGMP or MLD.";
        }
        leaf max-groups {
          if-feature global-max-groups;
          type uint32;
          description
            "The maximum number of groups that IGMP
             or MLD can join.";
        }
      } // global-config-attributes

      grouping global-state-attributes {

        description "Global IGMP and MLD state attributes.";

        leaf entries-count {
          type uint32;
          config false;
          description
            "The number of entries in IGMP or MLD.";
        }
        leaf groups-count {
          type uint32;
          config false;
          description
            "The number of groups that IGMP or MLD can join.";
        }

        container statistics {
          config false;
          description "Global statistics.";

          leaf discontinuity-time {
            type yang:date-and-time;
            description
              "The time on the most recent occasion at which any one
              or more of the statistic counters suffered a
              discontinuity. If no such discontinuities have occurred
              since the last re-initialization of the local
              management subsystem, then this node contains the time
              the local management subsystem re-initialized itself.";
          }
```

```
          container error {
            description "Statistics of errors.";
            uses global-statistics-error;
          }

          container received {
            description "Statistics of received messages.";
            uses global-statistics-sent-received;
          }
          container sent {
            description "Statistics of sent messages.";
            uses global-statistics-sent-received;
          }
        } // statistics
      } // global-state-attributes

      grouping global-statistics-error {
        description
          "A grouping defining statistics attributes for errors.";
        uses global-statistics-sent-received;
        leaf checksum {
          type yang:counter64;
          description
            "The number of checksum errors.";
        }
        leaf too-short {
          type yang:counter64;
          description
            "The number of messages that are too short.";
        }
      } // global-statistics-error

      grouping global-statistics-sent-received {
        description
          "A grouping defining statistics attributes.";
        leaf total {
          type yang:counter64;
          description
            "The number of total messages.";
        }
        leaf query {
          type yang:counter64;
          description
            "The number of query messages.";
        }
        leaf report {
          type yang:counter64;
          description
```

```
            "The number of report messages.";
```

```
        }
        leaf leave {
          type yang:counter64;
          description
            "The number of leave messages.";
        }
      } // global-statistics-sent-received

      grouping interface-global-config-attributes {
        description
          "Configuration attributes applied to the interface global level
           whose per interface attributes are not existing.";

        leaf max-groups-per-interface {
          if-feature intf-max-groups;
          type uint32;
          description
            "The maximum number of groups that IGMP or MLD can join.";
        }
      } //interface-global-config-attributes

      grouping interface-common-config-attributes {
        description
          "Configuration attributes applied to both the interface global
           level and interface level.";

        leaf last-member-query-interval {
          type uint16 {
            range "1..1023";
          }
          units seconds;
          default 1;
          description
            "Last Member Query Interval, which may be tuned to modify the
             leave latency of the network.";
           reference "RFC3376. Sec. 8.8.";
        }

        leaf query-interval {
          type uint16 {
            range "1..31744";
          }
          units seconds;
          default 125;
          description
            "The Query Interval is the interval between General Queries
             sent by the Querier.In RFC3376, Querier's Query Interval(QQI) is
             represented from the Querier's Query Interval Code in query
```

message as follows:

```
             If QQIC < 128, QQI = QQIC
             If QQIC >= 128, QQIC represents a floating-point value as follows:
              0 1 2 3 4 5 6 7
             +-+-+-+-+-+-+-+-+
             |1| exp | mant  |
             +-+-+-+-+-+-+-+-+
             QQI = (mant | 0x10) << (exp + 3)
             The maximum value of QQI is 31744.";
           reference "RFC3376. Sec. 4.1.7, 8.2, 8.14.2.";
         }

         leaf query-max-response-time {
           type uint16 {
             range "1..1023";
           }
           units seconds;
           default 10;
           description
             "Query maximum response time specifies the maximum time
              allowed before sending a responding report.";
           reference "RFC3376. Sec. 4.1.1, 8.3, 8.14.3.";
         }

         leaf require-router-alert {
           if-feature intf-require-router-alert;
           type boolean;
           default false;
           description
             "Protocol packets should contain router alert IP option.";
         }

         leaf robustness-variable {
           type uint8 {
             range "1..7";
           }
           default 2;
           description
             "Querier's Robustness Variable allows tuning for the expected
              packet loss on a network.";
           reference "RFC3376. Sec. 4.1.6, 8.1, 8.14.1.";
         }

       } // interface-common-config-attributes

       grouping interface-common-config-attributes-igmp {
         description
           "Configuration attributes applied to both the interface global
            level and interface level for IGMP.";
```

```
        uses interface-common-config-attributes;
```

```
        leaf version {
          type uint8 {
            range "1..3";
          }
          default 2;
          description "IGMP version.";
          reference "RFC1112, RFC2236, RFC3376.";
        }
      }

      grouping interface-common-config-attributes-mld {
        description
          "Configuration attributes applied to both the interface global
           level and interface level for MLD.";
        uses interface-common-config-attributes;
        leaf version {
          type uint8 {
            range "1..2";
          }
          default 2;
          description "MLD version.";
          reference "RFC2710, RFC3810.";
        }
      }

      grouping interfaces-config-attributes-igmp {
        description
          "Configuration attributes applied to the interface global
           level for IGMP.";
        uses interface-common-config-attributes-igmp;
        uses interface-global-config-attributes;
      }

      grouping interfaces-config-attributes-mld {
        description
          "Configuration attributes applied to the interface global
           level for MLD.";
        uses interface-common-config-attributes-mld;
        uses interface-global-config-attributes;
      }

      grouping interface-specific-config-attributes {
        description
          "Per interface configuration attributes for both IGMP and MLD
           whose are not existing in interface global level.";

        leaf enable {
          if-feature intf-admin-enable;
```

```
          type boolean;
```

```
          default false;
          description
            "true to enable IGMP or MLD on the interface;
             false to disable IGMP or MLD on the interface.";
        }
        leaf group-policy {
          type leafref {
            path "/acl:acls/acl:acl/acl:name";
          }
          description
            "Name of the access policy used to filter IGMP or MLD
             membership.A device can restrict the length
             and value of this name, possibly space and special
             characters are not allowed.";
        }
        leaf immediate-leave {
          if-feature intf-immediate-leave;
          type empty;
          description
            "If present, IGMP or MLD perform an immediate leave upon
             receiving an IGMPv2 or MLDv1 leave message.
             If the router is IGMP-enabled or MLD-enabled, it sends an
             IGMP or MLD last member query with a last member query
             response time. However, the router does not wait for
             the response time before it prunes off the group.";
        }
        leaf max-groups {
          if-feature intf-max-groups;
          type uint32;
          description
            "The maximum number of groups that IGMP ro MLD can join.";
        }
        leaf max-group-sources {
          if-feature intf-max-group-sources;
          type uint32;
          description
            "The maximum number of group sources.";
        }

        leaf source-policy {
          if-feature intf-source-policy;
          type leafref {
            path "/acl:acls/acl:acl/acl:name";
          }
          description
            "Name of the access policy used to filter sources.
             A device can restrict the length
             and value of this name, possibly space and special
```

```
              characters are not allowed.";
```

```
        }
        leaf verify-source-subnet {
          if-feature intf-verify-source-subnet;
          type empty;
          description
            "If present, the interface accepts packets with matching
             source IP subnet only.";
        }
        leaf explicit-tracking {
          if-feature intf-explicit-tracking;
          type empty;
          description
            "If present, IGMP/MLD-based explicit membership tracking function
             for multicast routers and IGMP/MLD proxy devices
             supporting IGMPv3/MLDv2. The explicit membership tracking
             function contributes to saving network resources and
             shortening leave latency.";
        }
        leaf exclude-lite {
          if-feature intf-exclude-lite;
          type empty;
          description
            "If present, lightweight IGMPv3 and MLDv2 protocols will run on the
             which simplify the standard versions of IGMPv3 and MLDv2.";
          reference "RFC5790";
        }

      } // interface-specific-config-attributes

      grouping interface-config-attributes-igmp {
        description
          "Per interface configuration attributes for IGMP.";

        uses interface-common-config-attributes-igmp;
        uses interface-specific-config-attributes;

        leaf-list join-group {
          if-feature intf-join-group;
          type rt-types:ipv4-multicast-group-address;
          description
            "The router joins this multicast group on the interface.";
        }

        list ssm-map {
          if-feature intf-ssm-map;
          key "ssm-map-source-addr ssm-map-group-policy";
          description "The policy for (*,G) mapping to (S,G).";
          leaf ssm-map-source-addr {
```

```
            type ssm-map-ipv4-addr-type;
```

```
            description
              "Multicast source IPv4 address.";
          }
          leaf ssm-map-group-policy {
            type string;
            description
              "Name of the policy used to define ssm-map rules.
               A device can restrict the length
               and value of this name, possibly space and special
               characters are not allowed. ";
          }
        }

        list static-group {
          if-feature intf-static-group;
          key "group-addr source-addr";
          description
            "A static multicast route, (*,G) or (S,G).";

          leaf group-addr {
            type rt-types:ipv4-multicast-group-address;
            description
              "Multicast group IPv4 address.";
          }
          leaf source-addr {
            type rt-types:ipv4-multicast-source-address;
            description
              "Multicast source IPv4 address.";
          }
        }
      } // interface-config-attributes-igmp



      grouping interface-config-attributes-mld {
        description
          "Per interface configuration attributes for MLD.";

        uses interface-common-config-attributes-mld;
        uses interface-specific-config-attributes;

        leaf-list join-group {
          if-feature intf-join-group;
          type rt-types:ipv6-multicast-group-address;
          description
            "The router joins this multicast group on the interface.";
        }

        list ssm-map {
```

```
          if-feature intf-ssm-map;
          key "ssm-map-source-addr ssm-map-group-policy";
          description "The policy for (*,G) mapping to (S,G).";
          leaf ssm-map-source-addr {
            type ssm-map-ipv6-addr-type;
            description
              "Multicast source IPv6 address.";
          }
          leaf ssm-map-group-policy {
            type string;
            description
              "Name of the policy used to define ssm-map rules.
               A device can restrict the length
               and value of this name, possibly space and special
               characters are not allowed.";
          }
        }

        list static-group {
          if-feature intf-static-group;
          key "group-addr source-addr";
          description
            "A static multicast route, (*,G) or (S,G).";

          leaf group-addr {
            type rt-types:ipv6-multicast-group-address;
            description
              "Multicast group IPv6 address.";
          }
          leaf source-addr {
            type rt-types:ipv6-multicast-source-address;
            description
              "Multicast source IPv6 address.";
          }
        }
      } // interface-config-attributes-mld

      grouping interface-state-attributes-igmp-mld {
        description
          "Per interface state attributes for both IGMP and MLD.";

        leaf oper-status {
          type enumeration {
            enum up {
              description
              "Ready to pass packets.";
            }
            enum down {
```

description

```
              "The interface does not pass any packets.";
            }
          }
          config false;
          mandatory true;
          description
            "Interface up or down state for IGMP or MLD protocol";
        }
      } // interface-config-attributes-igmp-mld

      grouping interface-state-attributes-igmp {

        description
          "Per interface state attributes for IGMP.";

        uses interface-state-attributes-igmp-mld;

        leaf querier {
          type inet:ipv4-address;
          config false;
          mandatory true;
          description "The querier address in the subnet";
        }
        leaf-list joined-group {
          if-feature intf-join-group;
          type rt-types:ipv4-multicast-group-address;
          config false;
          description
            "The routers that joined this multicast group.";
        }

        list group {
          key "group-address";
          config false;
          description
            "Multicast group membership information
            that joined on the interface.";

          leaf group-address {
            type rt-types:ipv4-multicast-group-address;
            description
              "Multicast group address.";
          }
          uses interface-state-group-attributes-igmp-mld;

          leaf last-reporter {
            type inet:ipv4-address;
            description
```

"The last host address which has sent the

```
               report to join the multicast group.";
          }
          list source {
            key "source-address";
            description
              "List of multicast source information
               of the multicast group.";

            leaf source-address {
              type inet:ipv4-address;
              description
                "Multicast source address in group record.";
            }
            uses interface-state-source-attributes-igmp-mld;
            leaf last-reporter {
              type inet:ipv4-address;
              description
                "The last host address which has sent the
                 report to join the multicast source and group.";
            }
            list host {
              if-feature intf-explicit-tracking;
              key "host-address";
              description
                "List of multicast membership hosts
                 of the specific multicast source-group.";

              leaf host-address {
                type inet:ipv4-address;
                description
                  "Multicast membership host address.";
              }
              uses interface-state-host-attributes-igmp-mld;
            }// list host
          } // list source
        } // list group
      } // interface-state-attributes-igmp

      grouping interface-state-attributes-mld {

        description
          "Per interface state attributes for MLD.";

        uses interface-state-attributes-igmp-mld;

        leaf querier {
          type inet:ipv6-address;
          config false;
```

```
          mandatory true;
```

```
          description
            "The querier address in the subnet.";
        }
        leaf-list joined-group {
          if-feature intf-join-group;
          type rt-types:ipv6-multicast-group-address;
          config false;
          description
            "The routers that joined this multicast group.";
        }

        list group {
          key "group-address";
          config false;
          description
            "Multicast group membership information
            that joined on the interface.";

          leaf group-address {
            type rt-types:ipv6-multicast-group-address;
            description
              "Multicast group address.";
          }
          uses interface-state-group-attributes-igmp-mld;
          leaf last-reporter {
            type inet:ipv6-address;
            description
              "The last host address which has sent the
               report to join the multicast group.";
          }
          list source {
            key "source-address";
            description
              "List of multicast source information
               of the multicast group.";

            leaf source-address {
              type inet:ipv6-address;
              description
                "Multicast source address in group record";
            }
            uses interface-state-source-attributes-igmp-mld;
            leaf last-reporter {
              type inet:ipv6-address;
              description
                "The last host address which has sent the
                 report to join the multicast source and group.";
            }
```

```
            list host {
```

```
              if-feature intf-explicit-tracking;
              key "host-address";
              description
                "List of multicast membership hosts
                 of the specific multicast source-group.";

              leaf host-address {
                type inet:ipv6-address;
                description
                  "Multicast membership host address.";
              }
              uses interface-state-host-attributes-igmp-mld;
            }// list host
          } // list source
        } // list group
      } // interface-state-attributes-mld

      grouping interface-state-group-attributes-igmp-mld {
        description
          "Per interface state attributes for both IGMP and MLD
           groups.";

        leaf expire {
          type uint32;
          units seconds;
          mandatory true;
          description
            "The time left before multicast group state expires.";
        }
        leaf filter-mode {
          type enumeration {
            enum "include" {
              description
                "In include mode, reception of packets sent
                 to the specified multicast address is requested
                 only from those IP source addresses listed in the
                 source-list parameter";
            }
            enum "exclude" {
              description
                "In exclude mode, reception of packets sent
                 to the given multicast address is requested
                 from all IP source addresses except those
                 listed in the source-list parameter.";
            }
          }
          mandatory true;
          description
```

"Filter mode for a multicast group,

```
             may be either include or exclude.";
        }
        leaf up-time {
          type uint32;
          units seconds;
          mandatory true;
          description
            "The elapsed time since the device created multicast group
record.";
        }
      } // interface-state-group-attributes-igmp-mld

      grouping interface-state-source-attributes-igmp-mld {
        description
          "Per interface state attributes for both IGMP and MLD
           source-group records.";

        leaf expire {
          type uint32;
          units seconds;
          mandatory true;
          description
            "The time left before multicast source-group state expires.";
        }
        leaf up-time {
          type uint32;
          units seconds;
          mandatory true;
          description
            "The elapsed time since the device created multicast
             source-group record.";
        }
        leaf host-count {
          if-feature intf-explicit-tracking;
          type uint32;
          description
            "The number of host addresses.";
        }
      } // interface-state-source-attributes-igmp-mld

      grouping interface-state-host-attributes-igmp-mld {
        description
          "Per interface state attributes for both IGMP and MLD
           hosts of source-group records.";

        leaf host-filter-mode {
          type enumeration {
            enum "include" {
```

```
                description
                  "In include mode";
```

```
            }
            enum "exclude" {
              description
                "In exclude mode.";
            }
          }
          mandatory true;
          description
            "Filter mode for a multicast membership
             host may be either include or exclude.";
        }
      }// interface-state-host-attributes-igmp-mld

      /*
       * Configuration and Operational state data nodes (NMDA version)
       */
      augment "/rt:routing/rt:control-plane-protocols"
      {
        description
          "IGMP augmentation to routing control plane protocol
           configuration and state.";

        container igmp {
          if-feature feature-igmp;
          description
            "IGMP configuration and operational state data.";

          container global {
            description
              "Global attributes.";
            uses global-config-attributes;
            uses global-state-attributes;
          }

          container interfaces {
            description
              "Containing a list of interfaces.";

            uses interfaces-config-attributes-igmp {
              if-feature global-interface-config;
            }

            list interface {
              key "interface-name";
              description
                "List of IGMP interfaces.";
              leaf interface-name {
                type if:interface-ref;
```

```
                must "/if:interfaces/if:interface[if:name = current()]/"
```

```
                 + "ip:ipv4" {
                description
                  "The interface must have IPv4 enabled.";
              }
              description
                "Reference to an entry in the global interface list.";
            }
            uses interface-config-attributes-igmp {
              if-feature per-interface-config;
            }
            uses interface-state-attributes-igmp;
          } // interface
        } // interfaces
      } // igmp
    }//augment

    augment "/rt:routing/rt:control-plane-protocols"
    {
      description
        "MLD augmentation to routing control plane protocol
         configuration and state.";

      container mld {
        if-feature feature-mld;
        description
          "MLD configuration and operational state data.";

        container global {
          description
            "Global attributes.";
          uses global-config-attributes;
          uses global-state-attributes;
        }

        container interfaces {
          description
            "Containing a list of interfaces.";

          uses interfaces-config-attributes-mld {
            if-feature global-interface-config;
          }

          list interface {
            key "interface-name";
            description
              "List of MLD interfaces.";
            leaf interface-name {
              type if:interface-ref;
```

```
                must "/if:interfaces/if:interface[if:name = current()]/"
```

```
                 + "ip:ipv6" {
                description
                  "The interface must have IPv6 enabled.";
              }
              description
                "Reference to an entry in the global interface list.";
            }
            uses interface-config-attributes-mld {
              if-feature per-interface-config;
            }
            uses interface-state-attributes-mld;
          } // interface
        } // interfaces
      } // mld
    } // augment

    /*
     * RPCs
     */
    rpc clear-igmp-groups {
      if-feature rpc-clear-groups;
      description
        "Clears the specified IGMP cache entries.";

      input {
        leaf interface-name {
          if-feature feature-igmp;
          type leafref {
            path "/rt:routing/rt:control-plane-protocols/"
              + "igmp-mld:igmp/igmp-mld:interfaces/"
              + "igmp-mld:interface/igmp-mld:interface-name";
          }
          description
            "Name of the IGMP interface.
             If it is not specified, groups from all interfaces are
             cleared.";
        }
        leaf group-address {
          type rt-types:ipv4-multicast-group-address;
          description
            "Multicast group IPv4 address.
             If it is not specified, all IGMP group entries are
             cleared.";
        }
        leaf source-address {
          type rt-types:ipv4-multicast-source-address;
          description
            "Multicast source IPv4 address.
```

If it is not specified, all IGMP source-group entries are

```
               cleared.";
          }
        }
      } // rpc clear-igmp-groups

      rpc clear-mld-groups {
        if-feature rpc-clear-groups;
        description
          "Clears the specified MLD cache entires.";

        input {
          leaf interface-name {
            if-feature feature-mld;
            type leafref {
              path "/rt:routing/rt:control-plane-protocols/"
                + "igmp-mld:mld/igmp-mld:interfaces/"
                + "igmp-mld:interface/igmp-mld:interface-name";
            }
            description
              "Name of the MLD interface.
               If it is not specified, groups from all interfaces are
               cleared.";
          }
          leaf group-addrss {
            type rt-types:ipv6-multicast-group-address;
            description
              "Multicast group IPv6 address.
               If it is not specified, all MLD group entries are
               cleared.";
          }
          leaf source-address {
            type rt-types:ipv6-multicast-source-address;
            description
              "Multicast source IPv6 address.
               If it is not specified, all MLD source-group entries are
               cleared.";
          }
        }
      } // rpc clear-mld-groups

      /*
       * Notifications
       */
    }
   <CODE ENDS>
```

5. Security Considerations

The YANG module specified in this document defines 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 [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

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:

igmp:global

This subtree specifies the configuration for the IGMP attributes at the global level on a device. Modifying the configuration can cause IGMP membership deleted or reconstructed on all the interfaces of a device.

igmp:interfaces

This subtree specifies the configuration for the IGMP attributes at all of the interfaces level on a device. Modifying the configuration can cause IGMP membership deleted or reconstructed on all the interfaces of a device.

igmp:interfaces/interface

This subtree specifies the configuration for the IGMP attributes at the interface level on a device. Modifying the configuration can cause IGMP membership deleted or reconstructed on a specific interface of a device.

These subtrees are all under

/rt:routing/rt:control-plane protocols/igmp:

mld:global

This subtree specifies the configuration for the MLD attributes at the global level on a device. Modifying the configuration can cause MLD membership deleted or reconstructed on all the interfaces of a device.

mld:interfaces

This subtree specifies the configuration for the MLD attributes at all of the interfaces level on a device. Modifying the configuration can cause MLD membership deleted or reconstructed on all the interfaces of a device.

mld:interfaces/interface

This subtree specifies the configuration for the MLD attributes at the interface level on a device. Modifying the configuration can cause MLD membership deleted or reconstructed on a specific interface of a device.

These subtrees are all under

/rt:routing/rt:control-plane-protocols/mld:

Unauthorized access to any data node of these subtrees can adversely affect the membership records of multicast routing subsystem on the local device. This may lead to network malfunctions, delivery of packets to inappropriate destinations, and other problems.

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/igmp

/rt:routing/rt:control-plane-protocols/mld

Unauthorized access to any data node of the above subtree can disclose the operational state information of IGMP or MLD on this device.

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

clear-igmp-groups

clear-mld-groups

Unauthorized access to any of the above RPC operations can delete the IGMP or MLD membership records on this device.

6. IANA Considerations

RFC Ed.: In this section, replace all occurrences of 'XXXX' with the actual RFC number (and remove this note).

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

URI: urn:ietf:params:xml:ns:yang:ietf-igmp-mld

Registrant Contact: The IESG.

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

This document registers the following YANG modules in the YANG Module Names registry [RFC7950]:

---

name: ietf-igmp-mld

namespace: urn:ietf:params:xml:ns:yang:ietf-igmp-mld

prefix: igmp-mld

reference: RFC XXXX

---

7. Acknowledgments

The authors would like to thank Steve Baillargeon, Hu Fangwei, Robert Kebler, Tanmoy Kundu, and Stig Venaas for their valuable contributions.

8. Contributing Authors

Yisong Liu
Huawei Technologies
Huawei Bldg., No.156 Beiqing Rd.
Beijing 100095
China

Email: liuyisong@huawei.com

9. References

9.1. Normative References

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.

[RFC2236] Fenner, W., "Internet Group Management Protocol, Version 2", RFC 2236, November 1997.

[RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, October 1999.

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.

[RFC3688] Mealling, M., "The IETF XML Registry", RFC 3688, January 2004

[RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, April 2012

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, July 2013

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, January 2017

[RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, December 2017

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, April 2018

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, April 2018

[RFC8344] M. Bjorklund, "A YANG Data Model for IP Management", RFC8344, April 2018

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, April 2018

[I-D.ietf-acl-yang] M. Jethanandani, L. Huang, S. Agarwal and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-19(work in progress), April 2018

9.2. Informative References

[RFC4541] M. Christensen, K. Kimball and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, May 2006.

[RFC4605] B. Fenner, H. He, B. Haberman, and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, August 2006.

[RFC5790] H. Liu, W. Cao and H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790, February 2010

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, April 2018

[I-D.ietf-netmod-rfc6087bis] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-20(work in progress), April 2018

Authors' Addresses

Xufeng Liu
Volta Networks

EMail: xufeng.liu.ietf@gmail.com

Feng Guo
Huawei Technologies
Huawei Bldg., No.156 Beiqing Rd.
Beijing 100095
China

Email: guofeng@huawei.com

Mahesh Sivakumar
Juniper Networks
1133 Innovation Way
Sunnyvale, California
USA

Email: sivakumar.mahesh@gmail.com

Pete McAllister
Metaswitch Networks
100 Church Street
Enfield EN2 6BQ
UK

EMail: pete.mcallister@metaswitch.com

Anish Peter
Individual

EMail: anish.ietf@gmail.com