PIM Working Group
INTERNET-DRAFT

Intended Status: Standard Track

X. Liu
Ericsson
F. Guo
Huawei
M. Sivakumar
Cisco
P. McAllister
Metaswitch Networks
A. Peter
Juniper Networks

Expires: September 20, 2016

March 20, 2016

A YANG data model for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD)
draft-guo-pim-igmp-mld-yang-00.txt

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.

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/1id-abstracts.html

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

Table of Contents

1. Introduction

YANG [RFC6020] [RFC6087] is a data definition language that was introduced to model the configuration and running state of a device managed using NETCONF [RFC6241]. YANG is now also being used as a component of wider management interfaces, such as CLIs.

This document defines a draft YANG data model that can be used to configure and manage Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) devices. Currently this model is incomplete, but it 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. 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 RFC 2119 [RFC2119].

1.2. Terminology

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

This draft employs YANG tree diagrams, which are explained in [I-D.ietf-netmod-rfc6087bis].

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 representation of some of extension features is not completely specified in this draft of the data model. This model is being circulated in its current form for early oversight and review of the basic hierarchy.

The operational state fields and rpcs of this model are also incomplete, though the structure of what has been written may be taken as representative of the structure of the model when complete.

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 covered by future Internet Drafts.

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 draft 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.

For the same reason, wide constant ranges (for example, timer maximum and minimum) will be used in the model. It is expected that vendors will augment the model with any specific restrictions that might be required. Vendors may also extend the features list with proprietary extensions.

2.3. Position of address family in hierarchy

The current draft 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 and MLD Configuration

The IGMP and MLD modules define the routing-instance-wide configuration options in a three-level hierarchy as listed below:

Global level: IGMP MLD configuration attributes for the entire routing instance

Interface-global: IGMP MLD configuration attributes applicable to all interfaces IGMP MLD configuration attributes applied to interfaces whose interface level attributes are not existing, with same attributes' value for those

Interface-level: IGMP MLD configuration 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. The module structure also applies, where applicable, to the operational state and notifications as well.

Our current direction is to agree to a routing-instance-centric (VRF) model as opposed to protocol-centric mainly because it fits well into the routing-instance model, and it is easier to map from the VRF-centric to the protocol-centric than the other way around due to forward references.

The IGMP and MLD model will augment "/rt:routing/rt:routing-instance/rt:routing-protocols:" as opposed to augmenting '' rt:routing-instance/rt:routing-protocols:/rt:routing-protocol" as the latter would allow multiple protocol instances per VRF, which does not make sense for IGMP and MLD.

```
augment /rt:routing/rt:routing-instance/rt:routing-protocols:
 +--rw igmp
 |  +--rw global
 |  |  +--rw enable?        boolean {global-admin-enable}?
 |  |  +--rw max-entries?   uint32 {global-max-entries}?
 |  |  +--rw max-groups?    uint32 {global-max-groups}?
 |  +--rw interfaces
 |     +--rw last-member-query-interval?   uint16
 |     +--rw max-groups-per-interface?     uint32 {intf-max-groups}?
 |     +--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 interface* [interface]
 |        +--rw interface                     if:interface-ref
 |        +--rw enable?                       boolean {intf-admin-enable}?
 |        +--rw group-policy?                 string
 |        +--rw immediate-leave?              empty {intf-immediate-leave}?
 |        +--rw last-member-query-interval?   uint16
 |        +--rw max-groups?                   uint32 {intf-max-groups}?
 |        +--rw max-group-sources?            uint32 {intf-max-group-
sources}?
 |        +--rw query-interval?               uint16
 |        +--rw query-max-response-time?      uint16
 |        +--rw require-router-alert?    s     boolean {intf-require-router-
alert}?
 |        +--rw robustness-variable?          uint8
 |        +--rw source-policy?                string {intf-source-policy}?
 |        +--rw verify-source-subnet?         empty {intf-verify-source-
```

```
subnet}?
    |        +--rw version?                      uint8
    |        +--rw join-group*                   inet:ipv4-address {intf-join-
group}?
    |        +--rw ssm-map* [source-addr group-policy] {intf-ssm-map}?
```

```
    |        |  +--rw source-addr     ssm-map-ipv4-addr-type
    |        |  +--rw group-policy    string
    |        +--rw static-group* [group source-addr] {intf-static-group}?
    |           +--rw group          inet:ipv4-address
    |           +--rw source-addr    source-ipv4-addr-type
    +--rw mld
       +--rw global
       |  +--rw enable?        boolean {global-admin-enable}?
       |  +--rw max-entries?   uint32 {global-max-entries}?
       |  +--rw max-groups?    uint32 {global-max-groups}?
       +--rw interfaces
          +--rw last-member-query-interval?   uint16
          +--rw max-groups-per-interface?     uint32 {intf-max-groups}?
          +--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 interface* [interface]
             +--rw interface                     if:interface-ref
             +--rw enable?                       boolean {intf-admin-enable}?
             +--rw group-policy?                 string
             +--rw immediate-leave?              empty {intf-immediate-leave}?
             +--rw last-member-query-interval?   uint16
             +--rw max-groups?                   uint32 {intf-max-groups}?
             +--rw max-group-sources?            uint32 {intf-max-group-
sources}?
             +--rw query-interval?               uint16
             +--rw query-max-response-time?      uint16
             +--rw require-router-alert?         boolean {intf-require-router-
alert}?
             +--rw robustness-variable?          uint8
             +--rw source-policy?                string {intf-source-policy}?
             +--rw verify-source-subnet?         empty {intf-verify-source-
subnet}?
             +--rw version?                      uint8
             +--rw join-group*                   inet:ipv6-address {intf-join-
group}?
             +--rw ssm-map* [source-addr group-policy] {intf-ssm-map}?
             |  +--rw source-addr     ssm-map-ipv6-addr-type
             |  +--rw group-policy    string
             +--rw static-group* [group source-addr] {intf-static-group}?
                +--rw group          inet:ipv6-address
                +--rw source-addr    source-ipv6-addr-type
```

3.2. IGMP and MLD Operational State

The IGMP and MLD module contains operational state information also in a three-level hierarchy as mentioned earlier.

Global level: IGMP MLD operational state attributes for the entire routing instance

Interface-global: IGMP MLD interface level operational state attributes applied to interfaces whose interface level attributes do not exist, with same attributes' value for those interfaces

```
      Interface-specific: IGMP MLD operational state attributes specific
   to the given interface.
   augment /rt:routing-state/rt:routing-instance/rt:routing-protocols:
    +--ro igmp
    |  +--ro global
    |  |  +--ro enable?          boolean {global-admin-enable}?
    |  |  +--ro max-entries?     uint32 {global-max-entries}?
    |  |  +--ro 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
    |  +--ro interfaces
    |     +--ro last-member-query-interval?   uint16
    |     +--ro max-groups-per-interface?     uint32 {intf-max-groups}?
    |     +--ro query-interval?               uint16
    |     +--ro query-max-response-time?      uint16
    |     +--ro require-router-alert?         boolean {intf-require-router-
alert}?
    |     +--ro robustness-variable?          uint8
    |     +--ro version?                      uint8
    |     +--ro interface* [interface]
    |        +--ro interface                     if:interface-ref
    |        +--ro enable?                       boolean {intf-admin-enable}?
    |        +--ro group-policy?                 string
    |        +--ro immediate-leave?              empty {intf-immediate-leave}?
    |        +--ro last-member-query-interval?   uint16
    |        +--ro max-groups?                   uint32 {intf-max-groups}?
    |        +--ro max-group-sources?            uint32 {intf-max-group-
sources}?
    |        +--ro query-interval?               uint16
    |        +--ro query-max-response-time?      uint16
    |        +--ro require-router-alert?         boolean {intf-require-router-
alert}?
```

```
    |       +--ro robustness-variable?          uint8
    |       +--ro source-policy?                string {intf-source-policy}?
    |       +--ro verify-source-subnet?         empty {intf-verify-source-
subnet}?
    |       +--ro version?                      uint8
    |       +--ro join-group*                   inet:ipv4-address {intf-join-
group}?
    |       +--ro ssm-map* [source-addr group-policy] {intf-ssm-map}?
```

```
    |        |  +--ro source-addr     ssm-map-ipv4-addr-type
    |        |  +--ro group-policy    string
    |        +--ro static-group* [group source-addr] {intf-static-group}?
    |        |  +--ro group          inet:ipv4-address
    |        |  +--ro source-addr    source-ipv4-addr-type
    |        +--ro oper-status?                  enumeration
    |        +--ro dr?                           inet:ipv4-address
    |        +--ro querier?                      inet:ipv4-address
    |        +--ro joined-group*                 inet:ipv4-address {intf-join-
group}?
    |        +--ro group* [address]
    |           +--ro address          inet:ipv4-address
    |           +--ro expire?          uint32
    |           +--ro filter-mode?     enumeration
    |           +--ro host-count?      uint32
    |           +--ro up-time?         uint32
    |           +--ro host*            inet:ipv4-address
    |           +--ro last-reporter?   inet:ipv4-address
    |           +--ro source* [address]
    |              +--ro address          inet:ipv4-address
    |              +--ro expire?          uint32
    |              +--ro up-time?         uint32
    |              +--ro last-reporter?   inet:ipv4-address
    +--ro mld
       +--ro global
       |  +--ro enable?          boolean {global-admin-enable}?
       |  +--ro max-entries?     uint32 {global-max-entries}?
       |  +--ro 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
       +--ro interfaces
          +--ro last-member-query-interval?   uint16
          +--ro max-groups-per-interface?     uint32 {intf-max-groups}?
```

```
          +--ro query-interval?               uint16
          +--ro query-max-response-time?      uint16
          +--ro require-router-alert?         boolean {intf-require-router-
alert}?
          +--ro robustness-variable?          uint8
          +--ro version?                      uint8
```

```
          +--ro interface* [interface]
             +--ro interface                    if:interface-ref
             +--ro enable?                      boolean {intf-admin-enable}?
             +--ro group-policy?                string
             +--ro immediate-leave?             empty {intf-immediate-leave}?
             +--ro last-member-query-interval?  uint16
             +--ro max-groups?                  uint32 {intf-max-groups}?
             +--ro max-group-sources?           uint32 {intf-max-group-
sources}?
             +--ro query-interval?              uint16
             +--ro query-max-response-time?     uint16
             +--ro require-router-alert?        boolean {intf-require-router-
alert}?
             +--ro robustness-variable?         uint8
             +--ro source-policy?               string {intf-source-policy}?
             +--ro verify-source-subnet?        empty {intf-verify-source-
subnet}?
             +--ro version?                     uint8
             +--ro join-group*                  inet:ipv6-address {intf-join-
group}?
             +--ro ssm-map* [source-addr group-policy] {intf-ssm-map}?
             |  +--ro source-addr     ssm-map-ipv6-addr-type
             |  +--ro group-policy    string
             +--ro static-group* [group source-addr] {intf-static-group}?
             |  +--ro group          inet:ipv6-address
             |  +--ro source-addr    source-ipv6-addr-type
             +--ro oper-status?                 enumeration
             +--ro querier?                     inet:ipv6-address
             +--ro joined-group*                inet:ipv6-address {intf-join-
group}?
             +--ro group* [address]
                +--ro address          inet:ipv6-address
                +--ro expire?          uint32
                +--ro filter-mode?     enumeration
                +--ro host-count?      uint32
                +--ro up-time?         uint32
                +--ro host*            inet:ipv6-address
                +--ro last-reporter?   inet:ipv6-address
                +--ro source* [address]
                   +--ro address          inet:ipv6-address
                   +--ro expire?          uint32
                   +--ro up-time?         uint32
                   +--ro last-reporter?   inet:ipv6-address
```

3.3. IGMP and MLD RPC

```
rpcs:
    +---x clear-igmp-groups {rpc-clear-groups}?
    |  +---w input
    |     +---w routing-instance?   rt:routing-instance-ref
    |     +---w interface?          leafref
```

```
    |     +---w group?              inet:ipv4-address
    +---x clear-mld-groups {rpc-clear-groups}?
       +---w input
          +---w routing-instance?   rt:routing-instance-ref
          +---w interface?          leafref
          +---w group?              inet:ipv4-address
```

4. IGMP and MLD YANG Modules

```
   module ietf-igmp-mld {
   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 {
     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>

      Editors:   ";

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

   revision 2016-03-01 {
     description
       "Initial revision.";
     reference
       "RFC XXXX: A YANG Data Model for IGMP";
   }

   /*
```

```
     * Features
     */
    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 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
```

```
   typedef source-ipv4-addr-type {
     type union {
       type enumeration {
```

```
        enum '*' {
          description
          "Any source address.";
        }
      }
      type inet:ipv4-address;
    }
    description
      "Multicast source IP address type.";
  } // source-ipv4-addr-type

  typedef source-ipv6-addr-type {
    type union {
      type enumeration {
        enum '*' {
          description
          "Any source address.";
        }
      }
      type inet:ipv6-address;
    }
    description
      "Multicast source IP address type.";
  } // 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;
      description
        "true to enable IGMP in the routing instance;
         false to disable IGMP in the routing instance.";
    }

    leaf max-entries {
      if-feature global-max-entries;
      type uint32;
      description
        "The maximum number of entries in IGMP.";
    }
    leaf max-groups {
      if-feature global-max-groups;
      type uint32;
```

```
      description
        "The maximum number of groups that IGMP can join.";
    }
  } // global-config-attributes
```

```
   grouping global-state-attributes {
     description "Global IGMP and MLD state attributes.";

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

     container statistics {
       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 interfaces-config-attributes {
     description
       "Configuration attributes applied to interfaces whose
        per interface attributes are not existing.";

     leaf last-member-query-interval {
       type uint16 {
         range "1..65535";
       }
       description
         "Last Member Query Interval, which may be tuned to modify the
          leave latency of the network.";
       reference "RFC3376. Sec. 8.8.";
     }
     leaf max-groups-per-interface {
       if-feature intf-max-groups;
       type uint32;
       description
         "The maximum number of groups that IGMP can join.";
     }
     leaf query-interval {
       type uint16;
```

```
       units seconds;
       default 125;
       description
         "The Query Interval is the interval between General Queries
```

```
          sent by the Querier.";
        reference "RFC3376. Sec. 4.1.7, 8.2, 8.14.2.";
      }
      leaf query-max-response-time {
        type uint16;
        units seconds;
        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;
        description
          "";
      }
      leaf robustness-variable {
        type uint8 {
          range "2..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.";
      }
      leaf version {
        type uint8 {
          range "1..3";
        }
        description "IGMP version.";
        reference "RFC1112, RFC2236, RFC3376.";
      }
    } // interfaces-config-attributes

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

      uses interface-config-attributes-igmp-mld;

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

      list ssm-map {
        if-feature intf-ssm-map;
        key "source-addr group-policy";
        description "";
```

```
        leaf source-addr {
          type ssm-map-ipv4-addr-type;
          description
            "Multicast source IP address.";
```

```
      }
      leaf group-policy {
        type string;
        description
          "Name of the access policy used to filter IGMP
           membership.";
      }
    }

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

      leaf group {
        type inet:ipv4-address;
        description
          "Multicast group IP address.";
      }
      leaf source-addr {
        type source-ipv4-addr-type;
        description
          "Multicast source IP address.";
      }
    }
  } // interface-config-attributes-igmp

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

    leaf enable {
      if-feature intf-admin-enable;
      type boolean;
      description
        "true to enable IGMP on the interface;
         false to disable IGMP on the interface.";
    }
    leaf group-policy {
      type string;
      description
        "Name of the access policy used to filter IGMP membership.";
    }
    leaf immediate-leave {
      if-feature intf-immediate-leave;
      type empty;
      description
        "If present, IGMP perform an immediate leave upon receiving an
         IGMP Version 2 (IGMPv2) leave message.
         If the router is IGMP-enabled, it sends an IGMP 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 last-member-query-interval {
```

```
        type uint16 {
          range "1..65535";
        }
        description
          "Last Member Query Interval, which may be tuned to modify the
           leave latency of the network.";
        reference "RFC3376. Sec. 8.8.";
      }
      leaf max-groups {
        if-feature intf-max-groups;
        type uint32;
        description
          "The maximum number of groups that IGMP can join.";
      }
      leaf max-group-sources {
        if-feature intf-max-group-sources;
        type uint32;
        description
          "The maximum number of group sources.";
      }
      leaf query-interval {
        type uint16;
        units seconds;
        default 125;
        description
          "The Query Interval is the interval between General Queries
           sent by the Querier.";
        reference "RFC3376. Sec. 4.1.7, 8.2, 8.14.2.";
      }
      leaf query-max-response-time {
        type uint16;
        units seconds;
        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;
        description
          "";
      }
      leaf robustness-variable {
        type uint8 {
          range "2..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.";
```

```
    }
    leaf source-policy {
      if-feature intf-source-policy;
      type string;
```

```
       description
         "Name of the access policy used to filter sources.";
     }
     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 version {
       type uint8 {
         range "1..3";
       }
       description "IGMP version.";
       reference "RFC1112, RFC2236, RFC3376.";
     }
   } // interface-config-attributes-igmp-mld

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

     uses interface-config-attributes-igmp-mld;

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

     list ssm-map {
       if-feature intf-ssm-map;
       key "source-addr group-policy";
       description "";
       leaf source-addr {
         type ssm-map-ipv6-addr-type;
         description
           "Multicast source IP address.";
       }
       leaf group-policy {
         type string;
         description
           "Name of the access policy used to filter IGMP
            membership.";
       }
     }

     list static-group {
       if-feature intf-static-group;
       key "group source-addr";
       description
```

```
          "A static multicast route, (*,G) or (S,G).";

        leaf group {
          type inet:ipv6-address;
```

```
        description
          "Multicast group IP address.";
      }
      leaf source-addr {
        type source-ipv6-addr-type;
        description
          "Multicast source IP address.";
      }
    }
  } // interface-config-attributes-mld

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

    uses interface-state-attributes-igmp-mld;

    leaf dr {
      type inet:ipv4-address;
      description "";
    }
    leaf querier {
      type inet:ipv4-address;
      description "";
    }
    leaf-list joined-group {
      if-feature intf-join-group;
      type inet:ipv4-address;
      description
        "The routers that joined this multicast group.";
    }

    list group {
      key "address";
      description "";

      leaf address {
        type inet:ipv4-address;
        description
          "";
      }
      uses interface-state-group-attributes-igmp-mld;
      leaf-list host {
        type inet:ipv4-address;
        description
          "";
      }
      leaf last-reporter {
        type inet:ipv4-address;
        description "";
      }
      list source {
```

```
          key "address";
          description "";

          leaf address {
```

```
          type inet:ipv4-address;
          description "";
        }
        uses interface-state-source-attributes-igmp-mld;
        leaf last-reporter {
          type inet:ipv4-address;
          description "";
        }
      } // list source
    } // list group
  } // interface-state-attributes-igmp

  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.";
        }
      }
      description "";
    }
  } // interface-config-attributes-igmp-mld

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

    uses interface-state-attributes-igmp-mld;

    leaf querier {
      type inet:ipv6-address;
      description "";
    }
    leaf-list joined-group {
      if-feature intf-join-group;
      type inet:ipv6-address;
      description
        "The routers that joined this multicast group.";
    }

    list group {
      key "address";
      description "";
```

```
        leaf address {
          type inet:ipv6-address;
          description
            "";
```

```
      }
      uses interface-state-group-attributes-igmp-mld;
      leaf-list host {
        type inet:ipv6-address;
        description
          "";
      }
      leaf last-reporter {
        type inet:ipv6-address;
        description "";
      }
      list source {
        key "address";
        description "";

        leaf address {
          type inet:ipv6-address;
          description "";
        }
        uses interface-state-source-attributes-igmp-mld;
        leaf last-reporter {
          type inet:ipv6-address;
          description "";
        }
      } // 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;
      description "";
    }
    leaf filter-mode {
      type enumeration {
        enum "include" {
          description
          "";
        }
        enum "exclude" {
          description
          "";
        }
      }
      description "";
    }
    leaf host-count {
```

```
        type uint32;
          description "";
      }
      leaf up-time {
```

```
       type uint32;
       units seconds;
       description "";
     }
   } // interface-state-group-attributes-igmp-mld

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

     leaf expire {
       type uint32;
       units seconds;
       description "";
     }
     leaf up-time {
       type uint32;
       units seconds;
       description "";
     }
   } // interface-state-source-attributes-igmp-mld

   /*
    * Configuration data nodes
    */
   augment "/rt:routing/rt:routing-instance/"
     + "rt:routing-protocols" {
     description
       "IGMP and MLD augmentation to routing instance configuration.";

     container igmp {
       description
         "IGMP configuration data.";

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

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

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

         list interface {
           key "interface";
           description
```

```
            "List of IGMP interfaces.";
          leaf interface {
            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;
            }
          } // interface
        } // interfaces
      } // igmp

      container mld {
        description
          "MLD configuration data.";

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

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

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

          list interface {
            key "interface";
            description
              "List of MLD interfaces.";
            leaf interface {
              type if:interface-ref;
              must "/if:interfaces/if:interface[if:name = current()]/"
                 + "ip:ipv6" {
                description
                  "The interface must have IPv4 enabled.";
              }
              description
                "Reference to an entry in the global interface
                 list.";
            }
            uses interface-config-attributes-mld {
              if-feature per-interface-config;
            }
          } // interface
```

```
      } // interfaces
    } // mld
  } // augment
```

```
   /*
    * Operational state data nodes
    */
   augment "/rt:routing-state/rt:routing-instance/"
     + "rt:routing-protocols" {
     description
       "IGMP and MLD augmentation to routing instance state.";

     container igmp {
       description
         "IGMP configuration 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 {
           if-feature global-interface-config;
         }

         list interface {
           key "interface";
           description
             "List of IGMP interfaces.";
           leaf interface {
             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

     container mld {
       description
```

```
          "MLD configuration 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 {
          if-feature global-interface-config;
        }

        list interface {
          key "interface";
          description
            "List of MLD interfaces.";
          leaf interface {
            type if:interface-ref;
            must "/if:interfaces/if:interface[if:name = current()]/"
               + "ip:ipv6" {
              description
                "The interface must have IPv4 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 tables.";

    input {
      leaf routing-instance {
        type rt:routing-instance-ref;
        description
          "Routing instance name identifying a specific routing
           instance.
           This leaf is optional for the rpc.
```

```
            If it is specified, the rpc will clear groups in the
            specified routing instance;
            if it is not specified, the rpc will clear all groups in
            all routing instances.";
```

```
       }
       leaf interface {
         type leafref {
           path "/rt:routing/rt:routing-instance"
             + "[rt:name=current()/../routing-instance]/"
             + "rt:routing-protocols/igmp/interfaces/interface/"
             + "interface";
         }
         description
           "Name of the IGMP interface.
            If it is not specified, groups from all interfaces are
            cleared.";
       }
       leaf group {
         type inet:ipv4-address;
         description
           "Multicast group IP address.
            If it is not specified, all IGMP group tables are
            cleared.";
       }
     }
   } // rpc clear-igmp-groups

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

     input {
       leaf routing-instance {
         type rt:routing-instance-ref;
         description
           "Routing instance name identifying a specific routing
            instance.
            This leaf is optional for the rpc.
            If it is specified, the rpc will clear groups in the
            specified routing instance;
            if it is not specified, the rpc will clear all groups in
            all routing instances.";
       }
       leaf interface {
         type leafref {
           path "/rt:routing/rt:routing-instance"
             + "[rt:name=current()/../routing-instance]/"
             + "rt:routing-protocols/mld/interfaces/interface/"
             + "interface";
         }
         description
           "Name of the MLD interface.
            If it is not specified, groups from all interfaces are
            cleared.";
       }
```

```
        leaf group {
          type inet:ipv4-address;
          description
            "Multicast group IP address.
```

```
            If it is not specified, all MLD group tables are
            cleared.";
        }
      }
    } // rpc clear-mld-groups

    /*
     * Notifications
     */
  }
```

4. Security Considerations

The data model defined does not introduce any security implications. This draft does not change any underlying security issues inherent in [I-D.ietf-netmod-routing-cfg].

5. IANA Considerations

TBD

6. Acknowledgements

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

7. References

7.1. Normative References

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

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

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

[I-D.ietf-netmod-rfc6087bis] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-05 (work in progress), October 2015.

7.2. Informative 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.

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

[RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, August 2006.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.

Authors' Addresses

Xufeng Liu
Ericsson
1595 Spring Hill Road, Suite 500
Vienna VA 22182
USA

EMail: xufeng.liu@ericsson.com

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

Email: guofeng@huawei.com

Mahesh Sivakumar
Cisco Systems, Inc.
510 McCarthy Blvd
Milpitas, California 95035
United States

Email: masivaku@cisco.com

Metaswitch Networks
100 Church Street
Enfield EN2 6BQ
UK

EMail: pete.mcallister@metaswitch.com

Anish Peter
Juniper Networks
Electra, Exora Business Park
Bangalore, KA 560103
India

EMail: anishp@juniper.net