

PIM Working Group  
Internet-Draft  
Intended Status: Standard Track  
Expires: December 14, 2019

X. Liu  
Volta Networks  
F. Guo  
Huawei  
M. Sivakumar  
Juniper  
P. McAllister  
Metaswitch Networks  
A. Peter  
Individual  
June 14, 2019

A YANG Data Model for Internet Group Management Protocol (IGMP) and  
Multicast Listener Discovery (MLD)  
draft-ietf-pim-igmp-mld-yang-15

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/lid-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 December 14, 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.

## Conventions used in this document

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

## Table of Contents

1. Introduction.....	3
1.1. Terminology.....	3
1.2. Tree Diagrams.....	4
1.3. Prefixes in Data Node Names.....	4
2. Design of Data model.....	4
2.1. Scope of Model.....	4
2.1.1. Parameters Not Covered at Global Level.....	5
2.1.2. Parameters Not Covered at Interface Level.....	5
2.2. Optional Capabilities.....	6
2.3. Position of Address Family in Hierarchy.....	6
3. Module Structure.....	7
3.1. IGMP Configuration and Operational State.....	7
3.2. MLD Configuration and Operational State.....	10
3.3. IGMP and MLD Actions.....	13
4. IGMP and MLD YANG Module.....	13
5. Security Considerations.....	43
6. IANA Considerations.....	45
7. Acknowledgments.....	46
8. Contributing Authors.....	46
9. References.....	46
9.1. Normative References.....	46
9.2. Informative References.....	48

## 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 command line interfaces (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. The protocol versions include IGMPv1 [RFC1112], IGMPv2 [RFC2236], IGMPv3 [RFC3376], MLDv1 [RFC2710], and MLDv2 [RFC3810]. The core features of the IGMP and MLD protocols are defined as required. Non-core features are defined as optional in the provided data model.

The YANG model in this document conforms to the Network Management Datastore Architecture (NMDA) [RFC8342].

### 1.1. Terminology

The terminology for describing YANG data models is found in [RFC6020] and [RFC7950], including:

- o augment
- o data model
- o data node
- o identity
- o module

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

IGMP:

Internet Group Management Protocol [RFC3376].

MLD:

Multicast Listener Discovery [RFC3810].

SSM:

Source-Specific Multicast service model [RFC3569] [RFC4607].

## 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 corresponding YANG module, as shown in Table 1.

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	[RFC8519]

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], MLDv1 [RFC2710], and MLDv2 [RFC3810].

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

This model can be used to configure and manage various versions of IGMP and MLD protocols. The operational state data and statistics can be retrieved by this model. Even though no protocol specific notifications are defined in this model, the subscription and push mechanism defined in [I-D.ietf-netconf-subscribed-notifications] and [I-D.ietf-netconf-yang-push] can be used by the user to subscribe to notifications on the data nodes in this model.

The model contains all the basic configuration parameters to operate the protocols listed above. Depending on the implementation choices, some systems may not allow some of the advanced parameters to be

configurable. The occasionally implemented parameters are modeled as optional features in this model, while the rarely implemented parameters are not included in this model and left for augmentation. This model can be extended, and has been structured in a way that such extensions can be conveniently made.

The protocol parameters covered in this model can be seen from the model structure described in Section 3.

The protocol parameters that were considered but are not covered in this model are described in the following sections.

#### 2.1.1. Parameters Not Covered at Global Level

The configuration parameters and operational states not covered on an IGMP instance or an MLD instance are:

- o Explicit tracking
- o Maximum transmit rate
- o Last member query count
- o Other querier present time
- o Send router alert
- o Startup query interval
- o Startup query count

#### 2.1.2. Parameters Not Covered at Interface Level

The configuration parameters and operational states not covered on an IGMP interface or an MLD interface are:

- o Disable router alert check
- o Drop IGMP version 1, IGMP version 2, or MLD version 1
- o Last member query count
- o Maximum number of sources
- o Other querier present time
- o Passive mode
- o Promiscuous mode

- o Query before immediate leave
- o Send router alert

## 2.2. Optional Capabilities

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

There is also value in widely-supported features being standardized, to provide a standardized way to access these features, to save work for individual vendors, and so that mapping between different vendors' configuration is not needlessly complicated. Therefore this model declares 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 protocol IGMP only supports IPv4, while the protocol MLD only supports IPv6. The data model defined in this document can be used for both IPv4 and IPv6 address families.

This document defines IGMP and MLD as separate schema branches in the structure. The benefits are:

- o The model can support IGMP (IPv4), MLD (IPv6), or both optionally and independently. Such flexibility cannot be achieved cleanly with a combined branch.
- o The structure is consistent with other YANG models such as RFC 8344, which uses separate branches for IPv4 and IPv6.

- o The separate branches for IGMP and MLD can accommodate their differences better and cleaner. The two branches can better support different features and node types.

### 3. Module Structure

This model augments the core routing data model specified in [RFC8349].

```

+--rw routing
  +--rw router-id?
  +--rw control-plane-protocols
    |   +--rw control-plane-protocol* [type name]
    |   |   +--rw type
    |   |   +--rw name
    |   |   +--rw igmp      <= Augmented by this Model
    |   |   ...
    |   |   +--rw mld       <= Augmented by this Model
    |   |   ...

```

The "igmp" container instantiates an IGMP protocol of version IGMPv1, IGMPv2, or IGMPv3. The "mld" container instantiates an MLD protocol of version MLDv1 or MLDv2.

The YANG data model defined in this document conforms to the Network Management Datastore Architecture (NMDA) [RFC8342]. The operational state data is combined with the associated configuration data in the same hierarchy [RFC8407].

A configuration data node is marked as mandatory only when its value must be provided by the user. Where nodes are not essential to protocol operation, they are marked as optional. Some other nodes are essential but have a default specified, so that they are also optional and need not be configured explicitly.

#### 3.1. IGMP Configuration and Operational State

The IGMP data is modeled as a schema subtree augmenting the "control-plane-protocol" data node under "/rt:routing/rt:control-plane-protocols" in the module ietf-routing, following the convention described in [RFC8349]. The augmentation to the module ietf-routing allows this model to support multiple instances of IGMP, but a restriction MAY be added depending on the implementation and the device. The identity "igmp" is derived from the "rt:control-plane-protocol" base identity and indicates that a control-plane-protocol instance is IGMP.

The IGMP subtree is a three-level hierarchy structure as listed below:

Global level: Including IGMP configuration and operational state attributes for the entire IGMP protocol instance in this router.

Interface-global level: Including configuration data nodes that are applicable to all the interfaces whose corresponding nodes are not defined or not configured at the interface level. For such a node at the interface level, the system uses the same value of the corresponding node at the interface-global level.

Interface level: Including IGMP configuration and operational state attributes specific to the given interface. For a configuration node at the interface level, there may exist a corresponding configuration node with the same name at the interface-global level. The value configured on a node at the interface level overrides the value configured on the corresponding node at the interface-global level.

```
augment /rt:routing/rt:control-plane-protocols
  /rt:control-plane-protocol:
    +--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 data is modeled as a schema subtree augmenting the "control-plane-protocol" data node under "/rt:routing/rt:control-plane-protocols" in the module ietf-routing, following the convention described in [RFC8349]. The augmentation to the module ietf-routing allows this model to support multiple instances of MLD, but a restriction MAY be added depending on the implementation and the device. The identity "mld" is derived from the "rt:control-plane-protocol" base identity and indicates that a control-plane-protocol instance is MLD.

The MLD subtree is a three-level hierarchy structure as listed below:

**Global level:** Including MLD configuration and operational state attributes for the entire MLD protocol instance in this router.

**Interface-global level:** Including configuration data nodes that are applicable to all the interfaces whose corresponding nodes are not defined or not configured at the interface level. For such a node at the interface level, the system uses the same value of the corresponding node at the interface-global level.

**Interface level:** Including MLD configuration and operational state attributes specific to the given interface. For a configuration node at the interface level, there may exist a

corresponding configuration node with the same name at the interface-global level. The value configured on a node at the interface level overrides the value configured on the corresponding node at the interface-global level.

```

augment /rt:routing/rt:control-plane-protocols
  /rt:control-plane-protocol:
    +--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 Actions

IGMP and MLD each have one action which clears the group membership cache entries for that protocol.

```

augment /rt:routing/rt:control-plane-protocols
  /rt:control-plane-protocol:
    +---rw igmp {feature-igmp}?
      +---x clear-groups {action-clear-groups}?
        +---w input
          +---w (interface)
            +---:(name)
            |   +---w interface-name?   leafref
            +---:(all)
              +---w all-interfaces?   empty
          +---w group-address           union
          +---w source-address
            rt-types:ipv4-multicast-source-address

augment /rt:routing/rt:control-plane-protocols
  /rt:control-plane-protocol:
    +---rw mld {feature-mld}?
      +---x clear-groups {action-clear-groups}?
        +---w input
          +---w (interface)
            +---:(name)
            |   +---w interface-name?   leafref
            +---:(all)
              +---w all-interfaces?   empty
          +---w group-address?          union
          +---w source-address?
            rt-types:ipv6-multicast-source-address

```

### 4. IGMP and MLD YANG Module

This module references [RFC1112], [RFC2236], [RFC2710], [RFC3376], [RFC3810], [RFC5790], [RFC6636], [RFC6991], [RFC8294], [RFC8343], [RFC8344], [RFC8349], and [RFC8519].

```
<CODE BEGINS> file "ietf-igmp-mld@2019-06-12.yang"
module ietf-igmp-mld {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-igmp-mld";
  prefix igmp-mld;

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

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

  import ietf-routing-types {
    prefix "rt-types";
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }

  import ietf-access-control-list {
    prefix "acl";
    reference
      "RFC 8519: YANG Data Model for Network Access Control Lists
      (ACLs)";
  }

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

  import ietf-interfaces {
    prefix "if";
    reference "RFC 8343: A YANG Data Model for Interface Management";
  }

  import ietf-ip {
    prefix ip;
    reference "RFC 8344: A YANG Data Model for IP Management";
  }

  organization
    "IETF PIM Working Group";
```

## contact

"WG Web: <<http://tools.ietf.org/wg/pim/>>  
WG List: <<mailto:pim@ietf.org>>  
  
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 the configuration and operational state for the Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) protocols.

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

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

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

```
// RFC Ed.: replace XXXX with actual RFC number and remove
// this note
revision 2019-06-12 {
  description
    "Initial revision.";
  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-max-entries {
  description
    "Support configuration of global max-entries.";
}

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

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

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

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

feature action-clear-groups {
  description
    "Support actions 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
     */
    identity igmp {
        base "rt:control-plane-protocol";
        description "IGMP protocol.";
        reference
            "RFC 3376: Internet Group Management Protocol, Version 3.";
    }

    identity mld {
        base "rt:control-plane-protocol";
        description "MLD protocol.";
        reference
            "RFC 3810: Multicast Listener Discovery Version 2 (MLDv2) for
            IPv6.";
    }
```

```
}

/*
 * Groupings
 */
grouping global-config-attributes {
  description
    "This grouping is used in either IGMP schema or MLD schema.
    When used in IGMP schema, this grouping contains the global
    configuration for IGMP;
    when used in MLD schema, this grouping contains the global
    configuration for MLD.";

  leaf enable {
    if-feature global-admin-enable;
    type boolean;
    default true;
    description
      "When this grouping is used for IGMP, this leaf indicates
      whether IGMP is enabled ('true') or disabled ('false')
      in the routing instance.
      When this grouping is used for MLD, this leaf indicates
      whether MLD is enabled ('true') or disabled ('false')
      in the routing instance.";
  }
  leaf max-entries {
    if-feature global-max-entries;
    type uint32;
    description
      "When this grouping is used for IGMP, this leaf indicates
      the maximum number of entries in the IGMP instance.
      When this grouping is used for MLD, this leaf indicates
      the maximum number of entries in the MLD instance.
      If this leaf is not specified, the number of entries is not
      limited.";
  }
  leaf max-groups {
    if-feature global-max-groups;
    type uint32;
    description
      "When this grouping is used for IGMP, this leaf indicates
      the maximum number of groups in the IGMP instance.
      When this grouping is used for MLD, this leaf indicates
      the maximum number of groups in the MLD instance.
      If this leaf is not specified, the number of groups is not
      limited.";
  }
} // global-config-attributes
```

```
grouping global-state-attributes {
  description
    "This grouping is used in either IGMP schema or MLD schema.
    When used in IGMP schema, this grouping contains the global
    IGMP state attributes;
    when used in MLD schema, this grouping contains the global
    MLD state attributes;";

  leaf entries-count {
    type uint32;
    config false;
    description
      "When this grouping is used for IGMP, this leaf indicates
      the number of entries in the IGMP instance.
      When this grouping is used for MLD, this leaf indicates
      the number of entries in the MLD instance.";
  }
  leaf groups-count {
    type uint32;
    config false;
    description
      "When this grouping is used for IGMP, this leaf indicates
      the number of existing groups in the IGMP instance.
      When this grouping is used for MLD, this leaf indicates
      the number of existing groups in the MLD instance.";
  }
}

container statistics {
  config false;
  description
    "When this grouping is used for IGMP, this container contains
    the statistics for the IGMP instance.
    When this grouping is used for MLD, this leaf indicates
    the statistics for the MLD instance.";

  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 configured.";

    leaf max-groups-per-interface {
        if-feature intf-max-groups;
        type uint32;
        description
            "The maximum number of groups associated with each interface.
            If this leaf is not specified, the number of groups is not
            limited.";
    }
} //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;
        description
            "When used in IGMP schema, this leaf indicates the Last
            Member Query Interval, which may be tuned to modify the
            leave latency of the network;
            when used in MLD schema, this leaf indicates the Last
            Listener Query Interval, which may be tuned to modify the
            leave latency of the network.
            This leaf is not applicable for version 1 of the IGMP. For
            version 2 and version 3 of the IGMP, and for all versions of
            the MLD, the default value of this leaf is 1.
            This leaf may be configured at the interface level or the
            interface-global level, with precedence given to the value
            at the interface level. If the leaf is not configured at
            either level, the default value is used.";
        reference
            "RFC 2236. Sec. 8.8. RFC 3376. Sec. 8.8.
            RFC 2710. Sec. 7.8. RFC 3810. Sec. 9.8.";
    }
}
leaf query-interval {
```

```

type uint16 {
    range "1..31744";
}
units seconds;
description
    "The Query Interval is the interval between General Queries
    sent by the Querier. In RFC 3376, the 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.
    The default value is 125.
    This leaf may be configured at the interface level or the
    interface-global level, with precedence given to the value
    at the interface level. If the leaf is not configured at
    either level, the default value is used.";
reference "RFC 3376. Sec. 4.1.7, 8.2, 8.14.2.";
}
leaf query-max-response-time {
    type uint16 {
        range "1..1023";
    }
    units seconds;
    description
        "Query maximum response time specifies the maximum time
        allowed before sending a responding report.
        The default value is 10.
        This leaf may be configured at the interface level or the
        interface-global level, with precedence given to the value
        at the interface level. If the leaf is not configured at
        either level, the default value is used.";
    reference "RFC 3376. Sec. 4.1.1, 8.3, 8.14.3.";
}
leaf require-router-alert {
    if-feature intf-require-router-alert;
    type boolean;
    description
        "Protocol packets should contain router alert IP option.
        When this leaf is not configured, the server uses the
        following rules to determine the operational value of this
        leaf:
        if this grouping is used in IGMP schema and the value of the

```

```
    leaf 'version' is 1, the value 'false' is operationally used
    by the server;
    if this grouping is used in IGMP schema and the value of the
    leaf 'version' is 2 or 3, the value 'true' is operationally
    used by the server;
    if this grouping is used in MLD schema, the value 'true' is
    operationally used by the server.
    This leaf may be configured at the interface level or the
    interface-global level, with precedence given to the value
    at the interface level. If the leaf is not configured at
    either level, the default value is used.";
  }
  leaf robustness-variable {
    type uint8 {
      range "1..7";
    }
    description
      "Querier's Robustness Variable allows tuning for the
      expected packet loss on a network.
      The default value is 2.
      This leaf may be configured at the interface level or the
      interface-global level, with precedence given to the value
      at the interface level. If the leaf is not configured at
      either level, the default value is used.";
    reference "RFC 3376. 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";
    }
    description
      "IGMP version.
      The default value is 2.
      This leaf may be configured at the interface level or the
      interface-global level, with precedence given to the value
      at the interface level. If the leaf is not configured at
      either level, the default value is used.";
    reference "RFC 1112, RFC 2236, RFC 3376.";
  }
}
```



```
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";
    }
    description
      "MLD version.
      The default value is 2.
      This leaf may be configured at the interface level or the
      interface-global level, with precedence given to the value
      at the interface level. If the leaf is not configured at
      either level, the default value is used.";
    reference "RFC 2710, RFC 3810.";
  }
}

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-level-config-attributes {
  description
    "This grouping is used in either IGMP schema or MLD schema.
    When used in IGMP schema, this grouping contains the IGMP
    configuration attributes that are defined at the interface
    level but are not defined at the interface-global level;
    when used in MLD schema, this grouping contains the MLD
    configuration attributes that are defined at the interface
    level but are not defined at the interface-global level.";
```

```
leaf enable {
  if-feature intf-admin-enable;
  type boolean;
  default true;
  description
    "When this grouping is used for IGMP, this leaf indicates
    whether IGMP is enabled ('true') or disabled ('false')
    on the interface.
    When this grouping is used for MLD, this leaf indicates
    whether MLD is enabled ('true') or disabled ('false')
    on the interface.";
}
leaf group-policy {
  type leafref {
    path "/acl:acls/acl:acl/acl:name";
  }
  description
    "When this grouping is used for IGMP, this leaf specifies
    the name of the access policy used to filter the
    IGMP membership.
    When this grouping is used for MLD, this leaf specifies
    the name of the access policy used to filter the
    MLD membership.
    The value space of this leaf is restricted to the existing
    policy instances defined by the referenced schema RFC 8519.
    As specified by RFC 8519, the length of the name is between
    1 and 64; a device MAY further restrict the length of this
    name; space and special characters are not allowed.
    If this leaf is not specified, no policy is applied, and
    all packets received from this interface are accepted.";
  reference
    "RFC 8519: YANG Data Model for Network Access Control Lists
    (ACLs)";
}
leaf immediate-leave {
  if-feature intf-immediate-leave;
  type empty;
  description
    "When this grouping is used for IGMP, the presence of this
    leaf requests IGMP to perform an immediate leave upon
    receiving an 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
    the group.
    When this grouping is used for MLD, the presence of this
    leaf requests MLD to perform an immediate leave upon
    receiving an MLDv1 leave message.
    If the router is MLD-enabled, it sends an MLD last member
```

```
        query with a last member query response time. However, the
        router does not wait for the response time before it prunes
        the group.";
    }
    leaf max-groups {
        if-feature intf-max-groups;
        type uint32;
        description
            "When this grouping is used for IGMP, this leaf indicates
            the maximum number of groups associated with the IGMP
            interface.
            When this grouping is used for MLD, this leaf indicates
            the maximum number of groups associated with the MLD
            interface.
            If this leaf is not specified, the number of groups is not
            limited.";
    }
    leaf max-group-sources {
        if-feature intf-max-group-sources;
        type uint32;
        description
            "The maximum number of group sources.
            If this leaf is not specified, the number of group sources
            is not limited.";
    }
    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.
            The value space of this leaf is restricted to the existing
            policy instances defined by the referenced schema RFC 8519.
            As specified by RFC 8519, the length of the name is between
            1 and 64; a device MAY further restrict the length of this
            name; space and special characters are not allowed.
            If this leaf is not specified, no policy is applied, and
            all packets received from this interface are accepted.";
    }
    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
  "When this grouping is used for IGMP, the presence of this
  leaf enables IGMP-based explicit membership tracking
  function for multicast routers and IGMP proxy devices
  supporting IGMPv3.
  When this grouping is used for MLD, the presence of this
  leaf enables MLD-based explicit membership tracking
  function for multicast routers and MLD proxy devices
  supporting MLDv2.
  The explicit membership tracking function contributes to
  saving network resources and shortening leave latency.";
reference
  "RFC 6636. Sec 3.";
}
leaf lite-exclude-filter {
  if-feature intf-lite-exclude-filter;
  type empty;
  description
    "When this grouping is used for IGMP, the presence of this
    leaf enables the support of the simplified EXCLUDE filter
    in the Lightweight IGMPv3 protocol, which simplifies the
    standard versions of IGMPv3.
    When this grouping is used for MLD, the presence of this
    leaf enables the support of the simplified EXCLUDE filter
    in the Lightweight MLDv2 protocol, which simplifies the
    standard versions of MLDv2.";
  reference "RFC 5790";
}
} // interface-level-config-attributes

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

  uses interface-common-config-attributes-igmp;
  uses interface-level-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).
     The version of IGMP must be 3 to support (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-level-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).
    The version of MLD must be 2 to support (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-mlld

grouping interface-state-attributes {
  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
        "Indicates whether the operational state of the interface
         is up or down.";
    }
} // interface-state-attributes

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

    uses interface-state-attributes;
    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;
    leaf last-reporter {
        type inet:ipv4-address;
        description
            "The IPv4 address of the last host 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;
    leaf last-reporter {
        type inet:ipv4-address;
        description
            "The IPv4 address of the last host 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 hosts with the membership for the specific
            multicast source-group.";

        leaf host-address {
            type inet:ipv4-address;
            description
                "The IPv4 address of the host.";
        }
        uses interface-state-host-attributes;
    } // list host
} // list source
} // list group
} // interface-state-attributes-igmp

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

    uses interface-state-attributes;
    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;
    leaf last-reporter {
      type inet:ipv6-address;
      description
        "The IPv6 address of the last host which has sent the
         report to join the multicast group.";
    }
  }
  list source {
    key "source-address";
    description
      "List of multicast sources of the multicast group.";

    leaf source-address {
      type inet:ipv6-address;
      description
        "Multicast source address in group record";
    }
    uses interface-state-source-attributes;
    leaf last-reporter {
      type inet:ipv6-address;
      description
        "The IPv6 address of the last host 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 hosts with the membership for the specific
       multicast source-group.";

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

```
        "The IPv6 address of the host.";
    }
    uses interface-state-host-attributes;
  } // list host
} // list source
} // list group
} // interface-state-attributes-mlld

grouping interface-state-group-attributes {
  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  
  
  grouping interface-state-source-attributes {  
    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  
  
  grouping interface-state-host-attributes {  
    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

/*
 * Configuration and Operational state data nodes (NMDA version)
 */
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol" {
    when "derived-from-or-self(rt:type, 'igmp-mld:igmp')" {
        description
            "This augmentation is only valid for a control-plane
            protocol instance of IGMP (type 'igmp').";
    }
    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 interface-global-config;
                refine query-interval {
                    default 125;
                }
                refine query-max-response-time {
                    default 10;
                }
                refine robustness-variable {
                    default 2;
                }
                refine version {
                    default 2;
                }
            }
        }
    }
}
```

```
}
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" {
      error-message
        "The interface must have IPv4 configured, either "
        + "enabled or disabled.";
    }
    description
      "Reference to an entry in the global interface list.";
  }
  uses interface-config-attributes-igmp {
    if-feature per-interface-config;
    refine last-member-query-interval {
      must "../version != 1 or "
        + "(not(..version) and "
        + "(../..version != 1 or not(../..version)))" {
      error-message
        "IGMPv1 does not support "
        + "last-member-query-interval.";
    }
  }
  refine max-group-sources {
    must "../version = 3 or "
      + "(not(..version) and (../..version = 3))" {
    error-message
      "The version of IGMP must be 3 to support the "
      + "source specific parameters.";
    }
  }
  refine source-policy {
    must "../version = 3 or "
      + "(not(..version) and (../..version = 3))" {
    error-message
      "The version of IGMP must be 3 to support the "
      + "source specific parameters.";
    }
  }
  refine explicit-tracking {
    must "../version = 3 or "
      + "(not(..version) and (../..version = 3))" {
    error-message
      "The version of IGMP must be 3 to support the "
```

```

        + "explicit tracking function.";
    }
}
refine lite-exclude-filter {
    must "../version = 3 or "
        + "(not(../version) and (../../version = 3))" {
        error-message
            "The version of IGMP must be 3 to support the "
            + "simplified EXCLUDE filter in the Lightweight "
            + "IGMPv3 protocol.";
    }
}
}
}
uses interface-state-attributes-igmp;
} // interface
} // interfaces

/*
 * Actions
 */
action clear-groups {
    if-feature action-clear-groups;
    description
        "Clears the specified IGMP cache entries.";

    input {
        choice interface {
            mandatory true;
            description
                "Indicates the interface(s) from which the cache
                entries are cleared.";
            case name {
                leaf interface-name {
                    type leafref {
                        path "/rt:routing/rt:control-plane-protocols/"
                            + "rt:control-plane-protocol/"
                            + "igmp-mld:igmp/igmp-mld:interfaces/"
                            + "igmp-mld:interface/igmp-mld:interface-name";
                    }
                    description
                        "Name of the IGMP interface.";
                }
            }
        }
        case all {
            leaf all-interfaces {
                type empty;
                description
                    "IGMP groups from all interfaces are cleared.";
            }
        }
    }
}

```

```

    }
  }
  leaf group-address {
    type union {
      type enumeration {
        enum '*' {
          description
            "Any group address.";
        }
      }
      type rt-types:ipv4-multicast-group-address;
    }
    mandatory true;
    description
      "Multicast group IPv4 address.
      If the value '*' is specified, all IGMP group entries
      are cleared.";
  }
  leaf source-address {
    type rt-types:ipv4-multicast-source-address;
    mandatory true;
    description
      "Multicast source IPv4 address.
      If the value '*' is specified, all IGMP source-group
      entries are cleared.";
  }
}
} // action clear-groups
} // igmp
} //augment

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol" {
  when "derived-from-or-self(rt:type, 'igmp-mld:mld')" {
    description
      "This augmentation is only valid for a control-plane
      protocol instance of IGMP (type 'mld').";
  }
  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 interface-global-config;
        refine last-member-query-interval {
            default 1;
        }
        refine query-interval {
            default 125;
        }
        refine query-max-response-time {
            default 10;
        }
        refine require-router-alert {
            default true;
        }
        refine robustness-variable {
            default 2;
        }
        refine version {
            default 2;
        }
    }
    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" {
                error-message
                    "The interface must have IPv6 configured, either "
                    + "enabled or disabled.";
            }
            description
                "Reference to an entry in the global interface list.";
        }
        uses interface-config-attributes-mld {
            if-feature per-interface-config;
            refine max-group-sources {
```



```

    must "../version = 2 or "
    + "(not(..version) and "
    + "(../..version = 2 or not(..../version)))" {
    error-message
        "The version of MLD must be 2 to support the "
        + "source specific parameters.";
    }
}
}
refine source-policy {
    must "../version = 2 or "
    + "(not(..version) and "
    + "(../..version = 2 or not(..../version)))" {
    error-message
        "The version of MLD must be 2 to support the "
        + "source specific parameters.";
    }
}
}
refine explicit-tracking {
    must "../version = 2 or "
    + "(not(..version) and "
    + "(../..version = 2 or not(..../version)))" {
    error-message
        "The version of MLD must be 2 to support the "
        + "explicit tracking function.";
    }
}
}
refine lite-exclude-filter {
    must "../version = 2 or "
    + "(not(..version) and "
    + "(../..version = 2 or not(..../version)))" {
    error-message
        "The version of MLD must be 2 to support the "
        + "simplified EXCLUDE filter in the Lightweight "
        + "MLDv2 protocol.";
    }
}
}
}
uses interface-state-attributes-mlld;
} // interface
} // interfaces

/*
 * Actions
 */
action clear-groups {
    if-feature action-clear-groups;
    description
        "Clears the specified MLD cache entries.";
}

```

```
input {
  choice interface {
    mandatory true;
    description
      "Indicates the interface(s) from which the cache
       entries are cleared.";
    case name {
      leaf interface-name {
        type leafref {
          path "/rt:routing/rt:control-plane-protocols/"
            + "rt:control-plane-protocol/"
            + "igmp-mld:mld/igmp-mld:interfaces/"
            + "igmp-mld:interface/igmp-mld:interface-name";
        }
        description
          "Name of the MLD interface.";
      }
    }
    case all {
      leaf all-interfaces {
        type empty;
        description
          "MLD groups from all interfaces are cleared.";
      }
    }
  }
  leaf group-address {
    type union {
      type enumeration {
        enum '*' {
          description
            "Any group address.";
        }
      }
      type rt-types:ipv6-multicast-group-address;
    }
    description
      "Multicast group IPv6 address.
       If the value '*' is specified, all MLD group entries
       are cleared.";
  }
  leaf source-address {
    type rt-types:ipv6-multicast-source-address;
    description
      "Multicast source IPv6 address.
       If the value '*' is specified, all MLD source-group
       entries are cleared.";
  }
}
```

```
    } // action clear-mld-groups
  } // mld
} // augment
}
<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 [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] 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:

Under /rt:routing/rt:control-plane-protocols  
/rt:control-plane-protocol/igmp-mld:igmp,

igmp-mld:global

This subtree specifies the configuration for the IGMP attributes at the global level on an IGMP instance. Modifying the configuration can cause IGMP membership to be deleted or reconstructed on all the interfaces of an IGMP instance.

igmp-mld:interfaces

This subtree specifies the configuration for the IGMP attributes at the interface-global level on a IGMP instance. Modifying the configuration can cause IGMP membership to be deleted or reconstructed on all the interfaces of an IGMP instance.

igmp-mld:interfaces/interface

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

```
Under /rt:routing/rt:control-plane-protocols
/rt:control-plane-protocol/igmp-mld:mld,
```

```
igmp-mld:global
```

This subtree specifies the configuration for the MLD attributes at the global level on an MLD instance. Modifying the configuration can cause MLD membership to be deleted or reconstructed on all the interfaces of an MLD instance.

```
igmp-mld:interfaces
```

This subtree specifies the configuration for the MLD attributes at the interface-global level on an MLD instance. Modifying the configuration can cause MLD membership to be deleted or reconstructed on all the interfaces of an MLD instance.

```
igmp-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 to be deleted or reconstructed on a specific interface of an MLD instance.

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
/rt:control-plane-protocol/igmmp-mld:igmp
```

```
/rt:routing/rt:control-plane-protocols
/rt:control-plane-protocol/igmp-mld: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 action 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:

```
/rt:routing/rt:control-plane-protocols  
/rt:control-plane-protocol/igmmp-mlld:igmp/igmmp-mlld:clear-groups
```

```
/rt:routing/rt:control-plane-protocols  
/rt:control-plane-protocol/igmp-mlld:mld/igmp-mlld:clear-groups
```

Unauthorized access to any of the above action 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-mlld

Registrant Contact: The IESG.

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

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

-----  
name: ietf-igmp-mlld  
namespace: urn:ietf:params:xml:ns:yang:ietf-igmp-mlld  
prefix: igmp-mlld  
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.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.

- [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.
- [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.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 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.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, March 2018.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, March 2018.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, March 2018.
- [RFC8344] M. Bjorklund, "A YANG Data Model for IP Management", RFC8344, March 2018.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, March 2018.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, August 2018.
- [RFC8519] M. Jethanandani, S. Agarwal, L. Huang and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, March 2019.

## 9.2. Informative References

- [RFC3569] Bhattacharyya, S., Ed., "An Overview of Source-Specific Multicast (SSM)", RFC 3569, July 2003.
- [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.
- [RFC6636] H. Asaeda, H. Liu and Q. Wu, "Tuning the Behavior of the Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) for Routers in Mobile and Wireless Networks", RFC 6636, May 2012.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, March 2018.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", RFC 8407, October 2018.
- [I-D.ietf-netconf-subscribed-notifications]  
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Customized Subscriptions to a Publisher's Event Streams", draft-ietf-netconf-subscribed-notifications-26 (work in progress), May 2019.
- [I-D.ietf-netconf-yang-push]  
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-25 (work in progress), May 2019.



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





PIM Working Group  
Internet-Draft  
Intended status: Informational  
Expires: May 13, 2019

LM. Contreras  
Telefonica  
CJ. Bernardos  
Universidad Carlos III de Madrid  
H. Asaeda  
NICT  
N. Leymann  
Deutsche Telekom  
November 9, 2018

Requirements for the extension of the IGMP/MLD proxy functionality to  
support multiple upstream interfaces  
draft-ietf-pim-multiple-upstreams-reqs-08

## Abstract

The purpose of this document is to define the requirements for a MLD (for IPv6) or IGMP (for IPv4) proxy with multiple interfaces covering a variety of applicability scenarios. The referred scenarios, while describing not sophisticated service situations, present cases that existing technology does not allow to solve in a simplistic manner. This document is then intended to serve as input for future documents defining the support of multiple upstream interfaces by IGMP/MLD proxies being compliant with the aforementioned requirements.

## Status of This Memo

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

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

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

This Internet-Draft will expire on May 13, 2019.

## Copyright Notice

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

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

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. Problem statement . . . . .	3
4. Scenarios of applicability . . . . .	5
4.1. Fixed network scenarios . . . . .	6
4.1.1. Multicast wholesale offer for residential services . . . . .	6
4.1.1.1. Requirements . . . . .	6
4.1.2. Multicast resiliency . . . . .	7
4.1.2.1. Requirements . . . . .	7
4.1.3. Load balancing for multicast traffic in the metro segment . . . . .	7
4.1.3.1. Requirements . . . . .	8
4.1.4. Network merging with different multicast services . . . . .	8
4.1.4.1. Requirements . . . . .	8
4.1.5. Multicast service migration . . . . .	9
4.1.5.1. Requirements . . . . .	9
4.2. Mobile network scenarios . . . . .	10
5. Summary of requirements . . . . .	10
6. Security Considerations . . . . .	11
7. IANA Considerations . . . . .	11
8. Acknowledgements . . . . .	12
9. References . . . . .	12
9.1. Normative References . . . . .	12
9.2. Informative References . . . . .	12
Authors' Addresses . . . . .	14

## 1. Introduction

The aim of this document is to define the functionality that an IGMP/MLD proxy with multiple upstream interfaces should have in order to support different scenarios of applicability in both fixed and mobile networks. IGMP/MLD proxies are a generic solution very much deployed in existing carrier networks. An extension to them in the sense of supporting multiple upstream interfaces can provide a more flexible and lightweight solution than other potential alternatives that could face more complexities (like multi-domain routing in the case of PIM,

or the need of some external elements -e.g., controllers- if the coordination of actions required lays outside the proxy).

The functional behavior of an IGMP/MLD proxy with multiple upstream interfaces here described is needed in order to simplify node functionality and to ensure an easier deployment of multicast capabilities in all the use cases described in this document.

For doing that, a number of scenarios are described, representing current deployments and needs from operator's networks. From that scenarios, certain requirements are identified as needed to simplify operational situations, enable optimized service delivery, etc. Those represent functional requirements to be satisfied by IGMP/MLD proxies with multiple upstream interfaces. These functional requirements reflect the need of coordinating actions from a single element in the network (i.e., the IGMP/MLD proxy), optimizing the delivery of the content within the network at any time.

Any Source Multicast (ASM) [RFC1112] and Source-Specific Multicast (SSM) [RFC4607] represent different service models at the time of subscribing to multicast groups by means of IGMPv3 [RFC3376], [RFC5790] and MLDv2 [RFC3810]. When using ASM a receiver joins a group indicating only the desired group address to be received. In the case of SSM, a receiver indicates the specific source address as well as a group address from where the multicast content is received. Both service models are taken into account along this document, and the specific requirements are derived from them.

## 2. Terminology

This document uses the terminology defined in [RFC4605]. Specifically, the definition of Upstream and Downstream interfaces, which are repeated here for completeness.

Upstream interface: A proxy device's interface in the direction of the root of the tree. Also called the "Host interface".

Downstream interface: Each of a proxy device's interfaces that is not in the direction of the root of the tree. Also called the "Router interfaces".

## 3. Problem statement

The concept of IGMP/MLD proxy with several upstream interfaces has emerged as a way of optimizing (and in some cases enabling) service delivery scenarios where separate multicast service providers are reachable through the same access network infrastructure. Figure 1 presents the conceptual model under consideration.

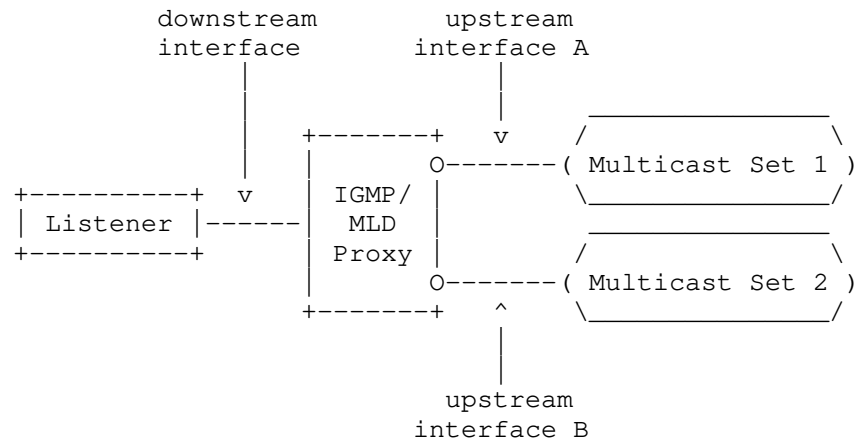


Figure 1: Concept of IGMP/MLD proxy with multiple upstream interfaces

This document is focused on both fixed and mobile network scenarios. Applicability of IGMP/MLD proxies with multiple upstream interfaces in mobile environments has been previously identified as beneficial in scenarios as the ones described in [RFC6224] and [RFC7287].

In the case of fixed networks, multicast wholesale services in a competitive residential market require an efficient distribution of multicast traffic from different operators or content providers, i.e. the incumbent operator and a number of alternative providers, on the network infrastructure of the former. Existing proposals are based on the use of PIM routing from the metro/core network, and multicast traffic aggregation on the same tree. A different approach could be achieved with the use of an IGMP/MLD proxy with multiple upstream interfaces, each of them pointing to a distinct multicast router in the metro/core border which is part of separated multicast trees deep in the network. Figure 2 graphically describes this scenario.

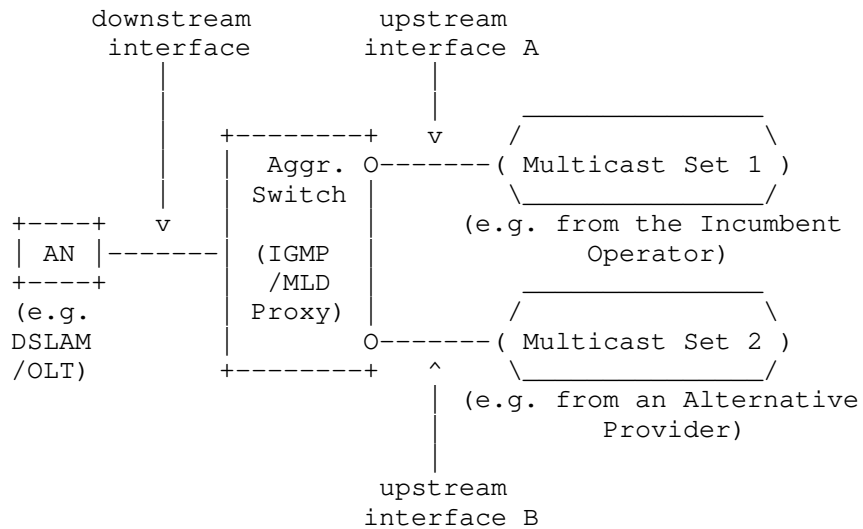


Figure 2: Example of usage of an IGMP/MLD proxy with multiple upstream interfaces in a fixed network scenario

Since those scenarios can motivate distinct needs in terms of IGMP/MLD proxy functionality, it is necessary to consider a comprehensive approach, looking at the possible scenarios, and establishing a minimum set of requirements which can allow the operation of a versatile IGMP/MLD proxy with multiple upstream interfaces as a common entity to all of them (i.e., no different kinds of proxies depending on the scenario, but a common proxy applicable to all the potential scenarios).

#### 4. Scenarios of applicability

Having multiple upstream interfaces creates a new decision space for delivering the proper multicast content to the subscriber. Basically it is now possible to implement channel-based (i.e., leveraging on multicast group IP address) or subscriber-based (i.e., referenced to the subscriber IP address) upstream selection, according to mechanisms or policies that could be defined for the multicast service provisioning.

This section describes in detail a number of scenarios of applicability of an IGMP/MLD proxy with multiple upstream interfaces in place. A number of requirements for the IGMP/MLD proxy functionality are identified from those scenarios.



All the exemplary scenarios here described are based on the support of two upstream interfaces. However, all of them are applicable also to the support of more than two upstream interfaces.

#### 4.1. Fixed network scenarios

Residential broadband users get access to multiple IP services through fixed network infrastructures. End user's equipment is connected to an access node, and the traffic of a number of access nodes is collected in aggregation switches.

For the multicast service, the use of an IGMP/MLD proxy with multiple upstream interfaces in those switches appears as a simple and straightforward solution.

##### 4.1.1. Multicast wholesale offer for residential services

This scenario has been already introduced in the previous section, and can be seen in Figure 2. There are two different operators, the one operating the fixed network where the end user is connected (e.g., typically an incumbent operator), and the one providing the Internet service to the end user (e.g., an alternative Internet service provider). Both can offer multicast streams that can be subscribed by the end user, independently of which provider contributes with the content.

Note that it is assumed that both providers offer distinct multicast groups. However, more than one subscription to multicast channels of different providers could take place simultaneously.

##### 4.1.1.1. Requirements

- o The IGMP/MLD proxy should be able to deliver multicast control messages sent by the end user to the corresponding provider's multicast router.
- o The IGMP/MLD proxy should be able to deliver multicast control messages sent by each of the providers to the corresponding end user.
- o The IGMP/MLD proxy should be able to support ASM and SSM at the time of requesting the content. Since the use case assumes that each provider offers distinct multicast groups, the IGMP/MLD proxy should be able to identify inconsistencies in the SSM requests, that is, the case in which for an (S, G) request the source S does not deliver a the group G.

#### 4.1.2. Multicast resiliency

In current PIM-based solutions [RFC7063], the resiliency of the multicast distribution relies on the routing capabilities provided by protocols like PIM [RFC7761] and VRRP [RFC5798]. A simpler scheme could be achieved by implementing different upstream interfaces on IGMP/MLD proxies, providing path diversity through the connection to distinct leaves of a given multicast tree.

It is assumed that only one of the upstream interfaces is active in receiving the multicast content, while the other is up and in standby mode for fast switching. The objective is to avoid video delivery affection that could imply play out interruption or buffering on the user side. Service parameters like the ones defined in [Y.1540] (such as packet loss ratio) or in [RFC4445] (like the delay factor) can be considered as parameters to be assessed from the service perspective. For instance, [TECH.3361-1] could be considered as a SLA framework to be satisfied in this case.

##### 4.1.2.1. Requirements

- o The IGMP/MLD proxy should be able to deliver multicast control messages received in the active upstream to the end users, while ignoring the control messages of the standby upstream interface.
- o The IGMP/MLD proxy should be able of rapidly switching from the active to the standby upstream interface in case of network failure, transparently to the end user.
- o The IGMP/MLD proxy should be able to deliver IGMP/MLD messages sent by the end user (for both ASM and SSM modes) to the corresponding active upstream interface.

#### 4.1.3. Load balancing for multicast traffic in the metro segment

A single upstream interface in existing IGMP/MLD proxy functionality [RFC4605] typically forces the distribution of all the channels on the same path in the last segment of the network. The metro and backhaul network is usually built using ring topologies. The devices in the ring implement IGMP/MLD functionality to join the content. Multiple upstream interfaces could naturally help to split the content demand, alleviating the bandwidth requirements in the overall metro segment by allowing some of the channels to follow the protection path, where spare capacity is vacant under normal conditions. This will allow, for instance, to absorb traffic peaks when a high number of channels (more than the expected on average) is requested.

#### 4.1.3.1. Requirements

- o The IGMP/MLD proxy should be able to deliver multicast control messages sent by the end user to the corresponding multicast router which provides the channel of interest.
- o The IGMP/MLD proxy should be able to deliver multicast control messages sent by each of the multicast routers to the corresponding end user.
- o The IGMP/MLD proxy should be able to decide which upstream interface is selected for any new channel request according to defined criteria (e.g., load balancing).
- o In the case of ASM, the IGMP/MLD proxy should be able to balance the traffic as a function of the group G requested. In the case of SSM, the load balancing mechanism could also consider the source S for the decision. In any case, the criteria will follow the policies defined by the network operator. Such policies can be influenced by the user requesting the service, for instance through the subscription to some channels being offered by a third party (which has reached an agreement with the provider for delivering that content in its network).

#### 4.1.4. Network merging with different multicast services

In some network merging situations, the multicast services provided before in each of the merged networks are maintained for the respective customer base (usually in a temporal fashion until the multicast service is redefined in a new single offer, but not necessarily, or not in short term, e.g. because of commercial agreements for each of the previous service offers).

In order to assist that network merging situations, IGMP/MLD proxies with multiple upstream interfaces can help in the transition simplifying the service provisioning and facilitating service continuity.

#### 4.1.4.1. Requirements

- o The IGMP/MLD proxy should be able to deliver multicast control messages sent by the end user to the corresponding multicast router which provides the channel of interest, according to the service subscription.
- o The IGMP/MLD proxy should be able to deliver multicast control messages sent by each of the multicast routers to the corresponding end user, according to the service subscription.

- o The IGMP/MLD proxy should be able to decide which upstream interface is selected for any new channel request according to defined criteria (e.g., service subscription).
- o For this use case, the usage of SSM can simplify the decision of the IGMP/MLD proxy. For ASM the decision should be assisted by further information like the service to which the end user is subscribed (e.g., taking into account what is the original network from where the end user was part previous to the network merge situation).

#### 4.1.5. Multicast service migration

This scenario considers the situation where a multicast service needs to be migrated from one upstream interface to another upstream interface (e.g. because of changes inside the service provider's network). The migration should be "smooth" and without any service interruption. In this case the multicast content is initially offered in both upstream interfaces and the proxy dynamically switches from the first to the second upstream interface, according to certain policies, and enabling to shut down the first upstream interface once the migration is completed.

##### 4.1.5.1. Requirements

- o The IGMP/MLD proxy should be able to deliver multicast control messages sent by the end user to the corresponding multicast router before and after the service migration.
- o The IGMP/MLD proxy should be able to deliver multicast control messages sent by each of the multicast routers to the corresponding end user, according to the situation of the user with respect to the service migration.
- o The IGMP/MLD proxy should be able to decide which upstream interface corresponds to each user, according to the situation of the user with respect to the service migration, i.e., the status of the user with respect the platform migration as purely operational situation while transitioning from one platform to another in a smooth manner.
- o The IGMP/MLD proxy should be able to decide which upstream interface corresponds to each ASM or SSM request, according to the situation of the group and source included in the request with respect to the service migration.

#### 4.2. Mobile network scenarios

Mobile networks offer different alternatives for multicast distribution.

One of them is defined by 3GPP [TS23.246] for the Multimedia Broadcast Multicast Service (MBMS). In this case, a MBMS gateway (MBMS GW) is connected to multiple evolved Node B (eNodeB) -- which are the base stations connecting the mobile handsets with the network wirelessly [TS36.300] -- for data distribution by means of IP multicast. The MBMS GW delivers the IP multicast groups. The eNodeB joins the appropriate group multicast address allocated by the MBMS GW to receive the content data. At this distribution level, an IGMP/MLD proxy could be part of the transport infrastructure providing connectivity to several distributed eNodeBs. The potential scenarios from this case do not essentially differentiate from the ones described for the fixed network scenarios, so the same situations and requirements apply.

Another alternative is given by Proxy Mobile IPv6 (PMIPv6) protocol for IP mobility management [RFC5213]. PMIPv6 is one of the mechanisms adopted by the 3GPP to support the mobility management of non-3GPP terminals in future Evolved Packet System (EPS) networks. PMIPv6 allows a Media Access Gateway (MAG) to establish a distinct bi-directional tunnel with different Local Mobility Anchors (LMAs), being each tunnel shared by the attached Mobile Nodes (MNs). Each mobile node is associated with a corresponding LMA, which keeps track of its current location, that is, the MAG where the mobile node is attached. As the basic solution for the distribution of multicast traffic within a PMIPv6 domain, [RFC6224] makes use of the bi-directional LMA-MAG tunnels. The use of an MLD proxy supporting multiple upstream interfaces can improve the performance and the scalability of multicast-capable PMIPv6 domains, for both multicast listener and multicast source mobility. Once again, the potential scenarios in this case are contained into the ones described for the fixed network scenarios, so the same situations and requirements apply.

#### 5. Summary of requirements

Following the analysis above, a number of different requirements can be identified by the IGMP/MLD proxy to support multiple upstream interfaces. The following table summarizes these requirements.

Functionality	Multicast Wholesale	Multicast Resiliency	Load Balancing	Network Merging	Network Migration
Upstream Control Delivery	X	X	X	X	X
Downstr. Control Delivery	X	X	X	X	X
Active / Standby Upstream		X			
Upstr i/f selection per group			X	X	
Upstr i/f selection all group		X			X
ASM	X	X	X	X	X
SSM	X	X	X		X

Figure 3: Functionality needed on IGMP/MLD proxy with multiple upstream interfaces per application scenario

## 6. Security Considerations

All the security considerations in [RFC4605] are directly applicable to this proposal.

## 7. IANA Considerations

There are no IANA considerations.

## 8. Acknowledgements

The authors would like to thank (in alphabetical order) Alvaro Retana, Thomas C. Schmidt, Stig Venaas and Dirk von Hugo for their comments and suggestions.

## 9. References

### 9.1. Normative References

- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <<https://www.rfc-editor.org/info/rfc1112>>.
- [RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, DOI 10.17487/RFC4605, August 2006, <<https://www.rfc-editor.org/info/rfc4605>>.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <<https://www.rfc-editor.org/info/rfc4607>>.
- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.

### 9.2. Informative References

- [RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <<https://www.rfc-editor.org/info/rfc3376>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.
- [RFC4445] Welch, J. and J. Clark, "A Proposed Media Delivery Index (MDI)", RFC 4445, DOI 10.17487/RFC4445, April 2006, <<https://www.rfc-editor.org/info/rfc4445>>.

- [RFC5213] Gundavelli, S., Ed., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, DOI 10.17487/RFC5213, August 2008, <<https://www.rfc-editor.org/info/rfc5213>>.
- [RFC5790] Liu, H., Cao, W., and H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790, DOI 10.17487/RFC5790, February 2010, <<https://www.rfc-editor.org/info/rfc5790>>.
- [RFC5798] Nadas, S., Ed., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, DOI 10.17487/RFC5798, March 2010, <<https://www.rfc-editor.org/info/rfc5798>>.
- [RFC6224] Schmidt, T., Waehlis, M., and S. Krishnan, "Base Deployment for Multicast Listener Support in Proxy Mobile IPv6 (PMIPv6) Domains", RFC 6224, DOI 10.17487/RFC6224, April 2011, <<https://www.rfc-editor.org/info/rfc6224>>.
- [RFC7063] Zheng, L., Zhang, J., and R. Parekh, "Survey Report on Protocol Independent Multicast - Sparse Mode (PIM-SM) Implementations and Deployments", RFC 7063, DOI 10.17487/RFC7063, December 2013, <<https://www.rfc-editor.org/info/rfc7063>>.
- [RFC7287] Schmidt, T., Ed., Gao, S., Zhang, H., and M. Waehlis, "Mobile Multicast Sender Support in Proxy Mobile IPv6 (PMIPv6) Domains", RFC 7287, DOI 10.17487/RFC7287, June 2014, <<https://www.rfc-editor.org/info/rfc7287>>.
- [TECH.3361-1] European Broadcasting Union, "Service Level Agreement for media transport services", EBU TECH.3361-1, September 2014.
- [TS23.246] "TS 23.246 Multimedia Broadcast/Multicast Service (MBMS); Architecture and functional description (Release 14) V14.1.0.", 3GPP TS 23.246 V14.1.0, December 2016.
- [TS36.300] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2", 3GPP TS 36.300 10.11.0, September 2013.



[Y.1540] ITU-T, "Internet protocol data communication service - IP packet transfer and availability performance parameters", ITU-T Y.1540, July 2016.

Authors' Addresses

Luis M. Contreras  
Telefonica  
Ronda de la Comunicacion, s/n  
Sur-3 building, 3rd floor  
Madrid 28050  
Spain

Email: [luismiguel.contrerasmurillo@telefonica.com](mailto:luismiguel.contrerasmurillo@telefonica.com)  
URI: <http://lmcontreras.com/>

Carlos J. Bernardos  
Universidad Carlos III de Madrid  
Av. Universidad, 30  
Leganes, Madrid 28911  
Spain

Phone: +34 91624 6236  
Email: [cjbc@it.uc3m.es](mailto:cjbc@it.uc3m.es)  
URI: <http://www.it.uc3m.es/cjbc/>

Hitoshi Asaeda  
National Institute of Information and Communications Technology  
4-2-1 Nukui-Kitamachi  
Koganei, Tokyo 184-8795  
Japan

Email: [asaeda@nict.go.jp](mailto:asaeda@nict.go.jp)

Nic Leymann  
Deutsche Telekom  
Germany

Email: [n.leymann@telekom.de](mailto:n.leymann@telekom.de)

Network Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: January 1, 2018

A. Gupta  
Avi Networks  
S. Venaas  
Cisco Systems  
June 30, 2017

PIM Encoding and Procedures for Unicast IPv4 prefixes with IPv6 next-hop  
draft-pim-with-ipv4-prefix-over-ipv6-nh-01.txt

## Abstract

Multi-Protocol BGP (MP-BGP) has support for distributing next-hop information for multiple address families using one AFI/SAFI Network Layer Reachability Information (NLRI). [RFC5549] specifies the extensions necessary to allow advertising IPv4 NLRI or VPN-IPv4 NLRI with a Next Hop address that belongs to the IPv6 protocol. While the next-hop info is learnt via MP-BGP, certain procedures are needed to enable traffic forwarding. This document describes PIM extensions and the use-cases for multicast forwarding in various scenarios.

## Status of This Memo

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

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

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

This Internet-Draft will expire on January 1, 2018.

## Copyright Notice

Copyright (c) 2017 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.

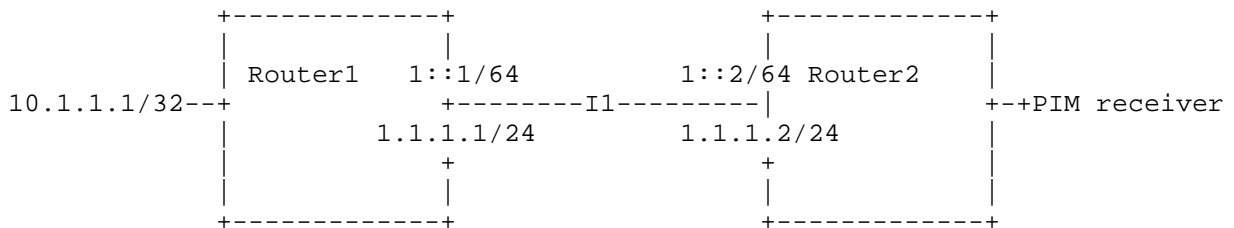
This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

1. Introduction . . . . .	2
2. Solution . . . . .	3
3. Security Considerations . . . . .	4
4. IANA Considerations . . . . .	4
5. References . . . . .	4
5.1. Normative References . . . . .	4
5.2. Informative References . . . . .	4
Authors' Addresses . . . . .	4

## 1. Introduction

Figure 1: Example Topology



While use of MP-BGP along with [RFC5549] enables one routing protocol session to exchange next-hop info for both IPv4 and IPv6 prefixes, forwarding plane needs additional procedures to enable forwarding in

data-plane. For example, when a IPv4 prefix is learnt over IPv6 next-hop, forwarding plane resolves the MAC-Address (L2-Adjacency) for IPv6 next-hop and uses it as destination-mac while doing inter-subnet forwarding. While it's simple to find the required information for unicast forwarding, multicast forwarding in same scenario poses additional requirements.

Multicast traffic is forwarding on a tree build by multicast routing protocols such as PIM. Multicast routing protocols are address family dependent and hence a system enabled with IPv4 and IPv6 multicast routing will have two PIM sessions one for each of the AF. Also, Multicast routing protocol uses Unicast reachability information to find unique Reverse Path Forwarding Neighbor. Further it sends control messages such as PIM Join to form the tree. Now when a PIMv4 session needs to initiate new multicast tree in event of discovering new receiver It consults Unicast control plane to find next-hop information. While this multicast tree can be Shared or Shortest Path tree, PIMv4 will need a PIMv4 neighbor to send join. However, the Unicast control plane can provide IPv6 next-hop as explained earlier and hence we need certain procedures to find corresponding PIMv4 neighbor address. This address is vital for correct prorogation of join and furthermore to build multicast tree. This document describes various approaches along with their use-cases and pros-cons.

In example topology, Router1 and Router2 are PIMv4 and PIMv6 neighbors on Interface I1. Router2 learns prefix 10.1.1.1/32's next-hop as 1::164 on Interface I1 as advertised by Router1 using BGP IPV6 NLRI. But in order to send (10.1.1.1/32, multicast-group) PIMv4 join on Interface I1, Router1 needs to find corresponding PIMv4 neighbor. In case there are multiple PIMv4 neighbors on same Interface I1, problem is aggravated.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

## 2. Solution

A PIM router can advertise its locally configured IPv6 addresses on the interface in PIMv4 Hello messages as per RFC4601 section 4.3.4. Same applies for IPv4 address in PIMv6 Hello. PIM will keep this info for each neighbor in Neighbor-cache along with DR-priority, hold-time etc. Once IPv6 Next-hop is notified to PIMv4, it will look into neighbors on the notified RPF-interface and find PIMv4 neighbor advertising same IPv6 local address in secondary Neighbor-list. If

such a match is found, that particular neighbor will be used as IPv4 RPF-Neighbor for initiating upstream join.

This method is valid for networks enabled with PIMv4 and PIMv6 both as well for the networks enabled with only PIMv4 with IPv6 BGP session or PIMv6 with IPv4 BGP session. This method doesn't require any additional config changes in the network.

### 3. Security Considerations

There are no new security considerations.

### 4. IANA Considerations

There are no IANA considerations.

### 5. References

#### 5.1. 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>>.

#### 5.2. Informative References

[RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, DOI 10.17487/RFC4601, August 2006, <<http://www.rfc-editor.org/info/rfc4601>>.

[RFC5549] Le Faucheur, F. and E. Rosen, "Advertising IPv4 Network Layer Reachability Information with an IPv6 Next Hop", RFC 5549, DOI 10.17487/RFC5549, May 2009, <<http://www.rfc-editor.org/info/rfc5549>>.

[RFC6395] Gulrajani, S. and S. Venaas, "An Interface Identifier (ID) Hello Option for PIM", RFC 6395, DOI 10.17487/RFC6395, October 2011, <<http://www.rfc-editor.org/info/rfc6395>>.

### Authors' Addresses

Ashutosh Gupta  
Avi Networks  
5155 Old Ironsides Dr. Suite 100  
Santa Clara, CA 95054  
USA

Email: ashutosh@avinetworks.com

Stig Venaas  
Cisco Systems  
821 Alder Drive  
San Jose, CA 95035  
USA

Email: stig@cisco.com

Internet Engineering Task Force  
Internet Draft  
Intended status: Standards Track  
Expires: April 2018

H. Zhao  
Ericsson  
X. Liu  
Jabil  
Y. Liu  
Huawei  
M. Sivakumar  
Cisco  
A. Peter  
Individual

October 26, 2017

A Yang Data Model for IGMP and MLD Snooping  
draft-zhao-pim-igmp-mld-snooping-yang-03.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) Snooping 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/ietf/lid-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 26, 2018.

## Copyright Notice

Copyright (c) 2017 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.

## Table of Contents

1. Introduction.....	2
1.1. Terminology.....	3
1.2. Tree Diagrams.....	3
2. Design of Data Model.....	3
2.1. Overview.....	4
2.2. IGMP and MLD Snooping Instances.....	4
2.3. IGMP and MLD Snooping References.....	10
2.4. IGMP and MLD Snooping RPC.....	13
3. IGMP and MLD Snooping YANG Module.....	13
4. Security Considerations.....	42
5. IANA Considerations.....	42
6. Normative References.....	42

## 1. Introduction

This document defines a YANG [RFC6020] data model for the management of Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping devices.

This data model follows the Guidelines for YANG Module Authors NMDA)[draft-dsdt-nmda-guidelines-01]. The "Network Management Datastore Architecture" (NMDA) adds the ability to inspect the current operational values for configuration, allowing clients to use identical paths for retrieving the configured values and the operational values.



### 1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119].

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

### 1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration (read-write), and "ro" means state data (read-only).
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "\*" denotes a list and leaf-list.
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

## 2. Design of Data Model

The model covers Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches [RFC4541].

The goal of this document is to define a data model that provides a common user interface to IGMP and MLD Snooping. There is very information that is designated as "mandatory", providing freedom for vendors to adapt this data model to their respective product implementations.

## 2.1. Overview

The IGMP and MLD Snooping YANG module defined in this document has all the common building blocks for the IGMP and MLD Snooping protocol.

The YANG module includes IGMP and MLD Snooping instances definition, instance references in the scenario of BRIDGE, VPLS. The module also includes the RPC methods for clearing the specified IGMP and MLD Snooping.

This YANG model follows the Guidelines for YANG Module Authors (NMDA) [draft-dsdt-nmda-guidelines-01]. This NMDA ("Network Management Datastore Architecture") architecture provides an architectural framework for datastores as they are used by network management protocols such as NETCONF [RFC6241], RESTCONF [RFC8040] and the YANG [RFC7950] data modeling language..

## 2.2. IGMP and MLD Snooping Instances

The YANG module defines IGMP and MLD Snooping instance. The instance will be referenced in all kinds of scenarios to configure IGMP and MLD Snooping. The attribute who could be read and written shows configuration data. The read-only attribute shows state data. The key attribute is name.

```
module: ietf-igmp-ml-d-snooping

  +--rw igmp-snooping-instances
  |   +--rw igmp-snooping-instance* [name]
  |       +--rw name                               string
  |       +--rw id?                                uint32
  |       +--rw type?                              enumeration
  |       +--rw enable?                            boolean {admin-enable}?
  |       +--rw forwarding-mode?                   enumeration
  |       +--rw explicit-tracking?                 boolean {explicit-tracking
}?
  |       +--rw exclude-lite?                     boolean {exclude-lite}?

```

```

    |      +---rw send-query?                               boolean
    |      +---rw fast-leave?                               empty {fast-leave}?
    |      +---rw last-member-query-interval?              uint16
    |      +---rw query-interval?                          uint16
    |      +---rw query-max-response-time?                uint16
    |      +---rw require-router-alert?                    boolean {require-router-al
ert}?
    |      +---rw robustness-variable?                     uint8
    |      +---rw version?                                 uint8
    |      +---rw static-bridge-mrouter-interface*         if:interface-ref {static-l
2-
    multicast-group}?
    |      +---rw static-vpls-mrouter-interface*           l2vpn-instance-pw-ref {sta
tic-
    l2-multicast-group}?
    |      +---rw querier-source?                          inet:ipv4-address
    |      +---rw static-l2-multicast-group* [group source-addr] {static-l2-
multicast-group}?
    |      |      +---rw group                              inet:ipv4-address
    |      |      +---rw source-addr                       source-ipv4-addr-type
    |      |      +---rw bridge-outgoing-interface*        if:interface-ref
    |      |      +---rw vpls-outgoing-ac*                 l2vpn-instance-ac-ref
    |      |      +---rw vpls-outgoing-pw*                 l2vpn-instance-pw-ref
    |      +---ro entries-count?                           uint32
    |      +---ro bridge-mrouter-interface*                if:interface-ref
    |      +---ro vpls-mrouter-interface*                  l2vpn-instance-pw-ref
    |      +---ro group* [address]
    |      |      +---ro address                          inet:ipv4-address

```

```

|   | +--ro mac-address?      yang:phys-address
|   | +--ro expire?          uint32
|   | +--ro up-time?         uint32
|   | +--ro last-reporter?   inet:ipv4-address
|   | +--ro source* [address]
|   |   +--ro address          inet:ipv4-address
|   |   +--ro bridge-outgoing-interface* if:interface-ref
|   |   +--ro vpls-outgoing-ac*  l2vpn-instance-ac-ref
|   |   +--ro vpls-outgoing-pw*  l2vpn-instance-pw-ref
|   |   +--ro up-time?         uint32
|   |   +--ro expire?         uint32
|   |   +--ro host-count?      uint32 {explicit-tracking}
?
|   |   +--ro last-reporter?   inet:ipv4-address
|   |   +--ro host* [host-address] {explicit-tracking}?
|   |     +--ro host-address    inet:ipv4-address
|   |     +--ro host-filter-mode? enumeration
|   +--ro statistics
|     +--ro received
|       | +--ro query?        yang:counter64
|       | +--ro membership-report-v1? yang:counter64
|       | +--ro membership-report-v2? yang:counter64
|       | +--ro membership-report-v3? yang:counter64
|       | +--ro leave?        yang:counter64
|       | +--ro pim?          yang:counter64

```

```

|      +--ro sent
|
|      +---ro query?                yang:counter64
|
|      +---ro membership-report-v1? yang:counter64
|
|      +---ro membership-report-v2? yang:counter64
|
|      +---ro membership-report-v3? yang:counter64
|
|      +---ro leave?                yang:counter64
|
|      +---ro pim?                  yang:counter64
+---rw mld-snooping-instances
|  +---rw mld-snooping-instance* [name]
|
|      +---rw name                    string
|
|      +---rw id?                    uint32
|
|      +---rw type?                  enumeration
|
|      +---rw enable?                boolean {admin-enable}?
|
|      +---rw forwarding-mode?       enumeration
|
|      +---rw explicit-tracking?     boolean {explicit-tracking
}|?
|
|      +---rw exclude-lite?          boolean {exclude-lite}?
|
|      +---rw send-query?            boolean
|
|      +---rw fast-leave?            empty {fast-leave}?
|
|      +---rw last-member-query-interval? uint16
|
|      +---rw query-interval?        uint16
|
|      +---rw query-max-response-time? uint16
|
|      +---rw require-router-alert?  boolean {require-router-al
}|?
|
|      +---rw robustness-variable?   uint8
|
|      +---rw version?               uint8

```

```

2-   |      +---rw static-bridge-mrouter-interface*   if:interface-ref {static-l
    multicast-group}?

    |      +---rw static-vpls-mrouter-interface*      l2vpn-instance-pw-ref {sta
tic-  l2-multicast-group}?

    |      +---rw querier-source?                     inet:ipv6-address

    |      +---rw static-l2-multicast-group* [group source-addr] {static-l2-
multicast-group}?

    |      |      +---rw group                        inet:ipv6-address
    |      |      +---rw source-addr                  source-ipv6-addr-type
    |      |      +---rw bridge-outgoing-interface*   if:interface-ref
    |      |      +---rw vpls-outgoing-ac*            l2vpn-instance-ac-ref
    |      |      +---rw vpls-outgoing-pw*           l2vpn-instance-pw-ref
    |      +---ro entries-count?                      uint32
    |      +---ro bridge-mrouter-interface*           if:interface-ref
    |      +---ro vpls-mrouter-interface*             l2vpn-instance-pw-ref
    |      +---ro group* [address]

    |      |      +---ro address                      inet:ipv6-address
    |      |      +---ro mac-address?                 yang:phys-address
    |      |      +---ro expire?                      uint32
    |      |      +---ro up-time?                    uint32
    |      |      +---ro last-reporter?              inet:ipv6-address
    |      |      +---ro source* [address]

    |      |      +---ro address                      inet:ipv6-address
    |      |      +---ro bridge-outgoing-interface*   if:interface-ref
    |      |      +---ro vpls-outgoing-ac*            l2vpn-instance-ac-ref

```

```

      |      |      +---ro vpls-outgoing-pw*          l2vpn-instance-pw-ref
      |      |      +---ro up-time?                  uint32
      |      |      +---ro expire?                    uint32
?    |      |      +---ro host-count?                  uint32 {explicit-tracking}
      |      |
      |      |      +---ro last-reporter?              inet:ipv6-address
      |      |      +---ro host* [host-address] {explicit-tracking}?
      |      |          +---ro host-address            inet:ipv6-address
      |      |          +---ro host-filter-mode?        enumeration
      |      +---ro statistics
      |          +---ro received
      |              |      +---ro query?              yang:counter64
      |              |      +---ro membership-report-v1? yang:counter64
      |              |      +---ro membership-report-v2? yang:counter64
      |              |      +---ro membership-report-v3? yang:counter64
      |              |      +---ro leave?              yang:counter64
      |              |      +---ro pim?                yang:counter64
      |          +---ro sent
      |              +---ro query?                    yang:counter64
      |              +---ro membership-report-v1?      yang:counter64
      |              +---ro membership-report-v2?      yang:counter64
      |              +---ro membership-report-v3?      yang:counter64
      |              +---ro leave?                    yang:counter64
      |              +---ro pim?                      yang:counter64

```

### 2.3. IGMP and MLD Snooping References

The IGMP and MLD Snooping instance could be referenced in the scenario of bridge, VPLS to configure the IGMP and MLD Snooping. The name of the instance is the key attribute.

The type of the instance indicates the scenario which is bridge or VPLS. When referenced in bridge, the id of instance means VLAN id. When referenced in VPLS, the id means VSI id.

```
module: ietf-igmp-mld-snooping
```

```
...
```

```
+--rw bridges
```

```
|   +--rw bridge* [name]
```

```
|       +--rw name                               dot1qtypes:name-type
```

```
|       +--rw igmp-snooping-instance?   igmp-snooping-instance-ref
```

```
|       +--rw mld-snooping-instance?   mld-snooping-instance-ref
```

```
|       +--rw component* [name]
```

```
|           +--rw name                     string
```

```
|           +--rw bridge-vlan
```

```
|               +--rw vlan* [vid]
```

```
|                   +--rw vid                               dot1qtypes:vlan-index-type
```

```
|                   +--rw igmp-snooping-instance?   igmp-snooping-instance-ref
```

```
|                   +--rw mld-snooping-instance?   mld-snooping-instance-ref
```

```
|                   +--rw interfaces
```

```
|                       +--rw interface* [name]
```

```
|                           +--rw name                     string
```

```
|                           +--rw igmp-snooping-instance?   igmp-snooping-instance-
ref
```



```

ef |                                     +-rw mld-snooping-instance?   mld-snooping-instance-r
+-rw l2vpn-instances
  +-rw l2vpn-instance* [name]
    +-rw name                               string
    +-rw igmp-snooping-instance?           igmp-snooping-instance-ref
    +-rw mld-snooping-instance?           mld-snooping-instance-ref
    +-rw endpoint* [name]
      +-rw name                             string
      +-rw igmp-snooping-instance?         igmp-snooping-instance-ref
      +-rw mld-snooping-instance?         mld-snooping-instance-ref
      +-rw (ac-or-pw-or-redundancy-grp)?
        +--:(ac)
          | +-rw ac* [name]
          |   +-rw name                     string
          |   +-rw igmp-snooping-instance? igmp-snooping-instance-ref
          |   +-rw mld-snooping-instance?  mld-snooping-instance-ref
        +--:(pw)
          | +-rw pw* [name]
          |   +-rw name                     string
          |   +-rw igmp-snooping-instance? igmp-snooping-instance-ref
          |   +-rw mld-snooping-instance?  mld-snooping-instance-ref
        +--:(redundancy-grp)
          +-rw (primary)
          | +--:(primary-ac)

```

```

| | +--rw primary-ac
| |   +--rw name? string
| |   +--rw igmp-snooping-instance? igmp-snooping-instan
ce-ref
| |   +--rw mld-snooping-instance? mld-snooping-instan
e-ref
| +--:(primary-pw)
|   +--rw primary-pw* [name]
|     +--rw name string
|     +--rw igmp-snooping-instance? igmp-snooping-instan
ce-ref
|     +--rw mld-snooping-instance? mld-snooping-instan
e-ref
+--rw (backup)?
  +--:(backup-ac)
  | +--rw backup-ac
  |   +--rw name? string
  |   +--rw igmp-snooping-instance? igmp-snooping-instan
ce-ref
  |   +--rw mld-snooping-instance? mld-snooping-instan
e-ref
  +--:(backup-pw)
  | +--rw backup-pw* [name]
  |   +--rw name string
  |   +--rw igmp-snooping-instance? igmp-snooping-instan
ce-ref
  |   +--rw mld-snooping-instance? mld-snooping-instan
e-ref

```

## 2.4. IGMP and MLD Snooping RPC

IGMP and MLD Snooping RPC clears the specified IGMP and MLD Snooping group tables.

```
rpcs:
  +---x clear-igmp-snooping-groups {rpc-clear-groups}?
  |   +---w input
  |   |   +---w id?          uint32
  |   |   +---w group?       inet:ipv4-address
  |   |   +---w source?      inet:ipv4-address
  +---x clear-mlD-snooping-groups {rpc-clear-groups}?
  |   +---w input
  |   |   +---w id?          uint32
  |   |   +---w group?       inet:ipv6-address
  |   |   +---w source?      inet:ipv6-address
```

## 3. IGMP and MLD Snooping YANG Module

```
<CODE BEGINS> file "ietf-igmp-mlD-snooping@2017-10-25.yang"
module ietf-igmp-mlD-snooping {
  namespace "urn:ietf:params:xml:ns:yang:ietf-igmp-mlD-snooping";
  // replace with IANA namespace when assigned
  prefix ims;

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

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

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-l2vpn {
    prefix "l2vpn";
  }

  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: Hongji Zhao  
<<mailto:hongji.zhao@ericsson.com>>

Xufeng Liu  
<[mailto:Xufeng\\_Liu@jabil.com](mailto:Xufeng_Liu@jabil.com)>

Yisong Liu  
<<mailto:liuyisong@huawei.com>>

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

Mahesh Sivakumar  
<<mailto:masivaku@cisco.com>>

";

description

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

revision 2017-10-25 {

description

"Change model definition to fit NMDA standard.";

reference

"RFC XXXX: A YANG Data Model for IGMP and MLD Snooping";

}

revision 2017-08-14 {

description

"using profile to cooperate with ieee-dot1Q-bridge module";

reference

"RFC XXXX: A YANG Data Model for IGMP and MLD Snooping";

}

revision 2017-06-28 {

description

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

```
    augment /rt:routing-state/rt:control-plane-protocols";
  reference
    "RFC XXXX: A YANG Data Model for IGMP and MLD Snooping";
}

revision 2017-02-05 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for IGMP and MLD Snooping";
}

/*
 * Features
 */

feature admin-enable {
  description
    "Support configuration to enable or disable IGMP and MLD
Snooping.";
}

feature fast-leave {
  description
    "Support configuration of fast-leave.";
}

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

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

feature static-l2-multicast-group {
  description
    "Support configuration of L2 multicast static-group.";
}

feature per-instance-config {
  description
    "Support configuration of each VLAN or VPLS instance or EVPN
instance.";
}
```

```
    feature rpc-clear-groups {
        description
            "Support to clear statistics by RPC for IGMP and MLD
Snooping.";
    }

    feature explicit-tracking {
        description
            "Support configuration of per instance explicit-tracking
hosts.";
    }

    feature exclude-lite {
        description
            "Support configuration of per instance exclude-lite.";
    }

    /*
     * Typedefs
     */
    typedef name-type {
        type string {
            length "0..32";
        }
        description
            "A text string of up to 32 characters, of locally determined
            significance.";
    }
    typedef vlan-index-type {
        type uint32 {
            range "1..4094 | 4096..4294967295";
        }
        description
            "A value used to index per-VLAN tables. Values of 0 and 4095
            are not permitted. The range of valid VLAN indices. If the
            value is greater than 4095, then it represents a VLAN with
            scope local to the particular agent, i.e., one without a
            global VLAN-ID assigned to it. Such VLANs are outside the
            scope of IEEE 802.1Q, but it is convenient to be able to
            manage them in the same way using this YANG module.";
        reference
            "IEEE Std 802.1Q-2014: Virtual Bridged Local Area Networks.";
    }
}
```

```
typedef igmp-snooping-instance-ref {
  type leafref {
    path "/igmp-snooping-instances/igmp-snooping-instance/name";
  }
  description
    "This type is used by data models that need to reference igmp
snooping instance.";
}

typedef mld-snooping-instance-ref {
  type leafref {
    path "/mld-snooping-instances/mld-snooping-instance/name";
  }
  description
    "This type is used by data models that need to reference mld
snooping instance.";
}

typedef l2vpn-instance-ac-ref {
  type leafref {
    path "/l2vpn:l2vpn/l2vpn:instances" +
        "/l2vpn:instance/l2vpn:endpoint/l2vpn:ac/l2vpn:name";
  }
  description "l2vpn-instance-ac-ref";
}

typedef l2vpn-instance-pw-ref {
  type leafref {
    path "/l2vpn:l2vpn/l2vpn:instances" +
        "/l2vpn:instance/l2vpn:endpoint/l2vpn:pw/l2vpn:name";
  }
  description "l2vpn-instance-pw-ref";
}

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 general-state-attributes {
    description "Statistics of IGMP and MLD Snooping ";

    container statistics {
        config false;
        description
            "The statistics of IGMP and MLD Snooping related packets.";

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



```
    grouping instance-config-attributes-igmp-snooping {
        description "IGMP snooping configuration for each VLAN or VPLS
instance or EVPN instance.";

        uses instance-config-attributes-igmp-mld-snooping;

    leaf querier-source {
        type inet:ipv4-address;
        description "Use the IGMP snooping querier to support IGMP
snooping in a VLAN where PIM and IGMP are not configured.
The IP address is used as the source address in
messages.";
    }

    list static-l2-multicast-group {
        if-feature static-l2-multicast-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.";
        }

        leaf-list bridge-outgoing-interface {
            when "ims:type = 'bridge'";
            type if:interface-ref;
            description "Outgoing interface in bridge fowarding";
        }

        leaf-list vpls-outgoing-ac {
            when "ims:type = 'vpls'";
            type l2vpn-instance-ac-ref;
            description "Outgoing ac in vpls fowarding";
        }
    }
```

```
    leaf-list vpls-outgoing-pw {
        when "ims:type = 'vpls'";
        type l2vpn-instance-pw-ref;
        description "Outgoing pw in vpls forwarding";
    }

} // static-l2-multicast-group

} // instance-config-attributes-igmp-snooping

grouping instance-config-attributes-igmp-mls-snooping {
    description
        "IGMP and MLD Snooping configuration of each VLAN.";

    leaf enable {
        if-feature admin-enable;
        type boolean;
        description
            "Set the value to true to enable IGMP and MLD Snooping in
the VLAN instance.";
    }

    leaf forwarding-mode {
        type enumeration {
            enum "mac" {
                description
                    "";
            }
            enum "ip" {
                description
                    "";
            }
        }
        description "The default forwarding mode for IGMP and MLD
Snooping is ip.
                    cisco command is as below
                    Router(config-vlan-config)# multicast snooping lookup
{ ip | mac } ";
    }

    leaf explicit-tracking {
        if-feature explicit-tracking;
        type boolean;
    }
}
```

```
    description "Tracks IGMP & MLD Snooping v3 membership reports
from individual hosts for each port of each VLAN or VSI.";
}

leaf exclude-lite {
    if-feature exclude-lite;
    type boolean;
    description
        "lightweight IGMPv3 and MLDv2 protocols, which simplify the
        standard versions of IGMPv3 and MLDv2.";
    reference "RFC5790";
}

leaf send-query {
    type boolean;
    default true;
    description "Enable quick response for topo changes.
        To support IGMP snooping in a VLAN where PIM and IGMP are
not configured.
        It cooperates with param querier-source. ";
}

/**
leaf mrouter-aging-time {
    type uint16 ;
    default 180;
    description "Aging time for mrouter interface";
}
**/

leaf fast-leave {
    if-feature fast-leave;
    type empty;
    description
        "When fast leave is enabled, the IGMP software assumes that
no more than one host is present on each VLAN port.";
}

leaf last-member-query-interval {
    type uint16 {
        range "1..65535";
    }
    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;
        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;
        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 require-router-alert;
        type boolean;
        default false;
        description
            "When the value is true, router alert exists in the IP head
of IGMP or MLD packet.";
    }

    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 and MLD Snooping version.";
    }

    leaf-list static-bridge-mrouter-interface {

      when "ims:type = 'bridge'";
      if-feature static-l2-multicast-group;
      type if:interface-ref;
      description "static mrouter interface in bridge forwarding";
    }

    leaf-list static-vpls-mrouter-interface {

      when "ims:type = 'vpls'";
      if-feature static-l2-multicast-group;
      type l2vpn-instance-pw-ref;
      description "static mrouter interface in vpls forwarding";
    }

  } // instance-config-attributes-igmp-ml-d-snooping

  grouping instance-config-attributes-ml-d-snooping {
    description "MLD snooping configuration of each VLAN.";

    uses instance-config-attributes-igmp-ml-d-snooping;

    leaf querier-source {
      type inet:ipv6-address;
      description
        "Use the MLD snooping querier to support MLD snooping where PIM
and MLD are not configured.
        The IP address is used as the source address in messages.";
    }

    list static-l2-multicast-group {
      if-feature static-l2-multicast-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.";
    }

    leaf-list bridge-outgoing-interface {
        when "ims:type = 'bridge'";
        type if:interface-ref;
        description "Outgoing interface in bridge forwarding";
    }

    leaf-list vpls-outgoing-ac {
        when "ims:type = 'vpls'";
        type l2vpn-instance-ac-ref;
        description "Outgoing ac in vpls forwarding";
    }

    leaf-list vpls-outgoing-pw {
        when "ims:type = 'vpls'";
        type l2vpn-instance-pw-ref;
        description "Outgoing pw in vpls forwarding";
    }

} // static-l2-multicast-group

} // instance-config-attributes-mld-snooping

grouping instance-state-group-attributes-igmp-mld-snooping {
    description
        "Attributes for both IGMP and MLD snooping groups.";

    leaf mac-address {
```

```
        type yang:phys-address;
        description "Destination mac address for L2 multicast
forwarding.";
    }

    leaf expire {
        type uint32;
        units seconds;
        description
            "The time left before multicast group timeout.";
    }

    leaf up-time {
        type uint32;
        units seconds;
        description
            "The time after the device created L2 multicast record.";
    }

} // instance-state-group-attributes-igmp-mld-snooping

grouping instance-state-attributes-igmp-snooping {
    description
        "State attributes for IGMP snooping for each VLAN or VPLS
instance or EVPN instance.";

    uses instance-state-attributes-igmp-mld-snooping;

    list group {
        key "address";
        config false;

        description "IGMP snooping information";

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

    }

    uses instance-state-group-attributes-igmp-mld-snooping;
}
```

```
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 "address";
  description "Source IP address for multicast stream";
  leaf address {
    type inet:ipv4-address;
    description "Source IP address for multicast stream";
  }

  uses instance-state-source-attributes-igmp-mld-snooping;
}

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 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.";
  }
  leaf host-filter-mode {
    type enumeration {
      enum "include" {
        description
          "In include mode";
      }
      enum "exclude" {
        description
          "In exclude mode.";
      }
    }
  }
}
```



```
        description
        "Filter mode for a multicast membership
        host may be either include or exclude.";
    }
} // list host

    } // list source
} // list group

// statistics
uses general-state-attributes;

} // instance-state-attributes-igmp-snooping

grouping instance-state-attributes-igmp-mlD-snooping {

    description
    "State attributes for both IGMP and MLD Snooping of each
VLAN or VPLS instance or EVPN instance.";

    leaf entries-count {
        type uint32;
        config false;
        description
        "The number of L2 multicast entries in IGMP and MLD
Snooping.";
    }

    leaf-list bridge-mrouter-interface {

        when "ims:type = 'bridge'";
        type if:interface-ref;
        config false;
        description " mrouter interface in bridge forwarding";

    }

    leaf-list vpls-mrouter-interface {

        when "ims:type = 'vpls'";
        type l2vpn-instance-pw-ref;
        config false;
        description " mrouter interface in vpls forwarding";

    }

}
```

```
    }

} // instance-config-attributes-igmp-mld-snooping

grouping instance-state-attributes-mld-snooping {
  description
    "State attributes for MLD snooping of each VLAN.";

  uses instance-state-attributes-igmp-mld-snooping;

  list group {
    key "address";

    config false;

    description "MLD snooping statistics information";

    leaf address {
      type inet:ipv6-address;
      description
        "Multicast group IP address";
    }

    uses instance-state-group-attributes-igmp-mld-snooping;

    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 "address";
      description "Source IP address for multicast stream";

      leaf address {
        type inet:ipv6-address;
        description "Source IP address for multicast stream";
      }

      uses instance-state-source-attributes-igmp-mld-snooping;

      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 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.";
    }
    leaf host-filter-mode {
        type enumeration {
            enum "include" {
                description
                    "In include mode";
            }
            enum "exclude" {
                description
                    "In exclude mode.";
            }
        }
        description
            "Filter mode for a multicast membership
host may be either include or exclude.";
    }
} // list host

} // list source
} // list group

// statistics
uses general-state-attributes;

} // instance-state-attributes-mld-snooping

grouping instance-state-source-attributes-igmp-mld-snooping {
    description
        "State attributes for both IGMP and MLD Snooping of each VLAN
or VPLS instance or EVPN instance.";
```

```
leaf-list bridge-outgoing-interface {
  when "ims:type = 'bridge'";
  type if:interface-ref;
  description "Outgoing interface in bridge forwarding";
}

leaf-list vpls-outgoing-ac {
  when "ims:type = 'vpls'";
  type l2vpn-instance-ac-ref;
  description "Outgoing ac in vpls forwarding";
}

leaf-list vpls-outgoing-pw {
  when "ims:type = 'vpls'";
  type l2vpn-instance-pw-ref;
  description "Outgoing pw in vpls forwarding";
}

leaf up-time {
  type uint32;
  units seconds;
  description "The time after the device created L2 multicast
record";
}

leaf expire {
  type uint32;
  units seconds;
  description
    "The time left before multicast group timeout.";
}

leaf host-count {
  if-feature explicit-tracking;
  type uint32;
  description
    "The number of host addresses.";
}

} // instance-state-source-attributes-igmp-mld-snooping

grouping general-statistics-error {
  description
```

```
    "A grouping defining statistics attributes for errors.";

    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.";
    }
} // general-statistics-error

grouping general-statistics-sent-received {
    description
        "A grouping defining statistics attributes.";

    leaf query {
        type yang:counter64;
        description
            "The number of query messages.";
    }
    leaf membership-report-v1 {
        type yang:counter64;
        description
            "The number of membership report v1 messages.";
    }
    leaf membership-report-v2 {
        type yang:counter64;
        description
            "The number of membership report v2 messages.";
    }
    leaf membership-report-v3 {
        type yang:counter64;
        description
            "The number of membership report v3 messages.";
    }
    leaf leave {
        type yang:counter64;
        description
            "The number of leave messages.";
    }
    leaf pim {
        type yang:counter64;
        description
            "The number of pim hello messages.";
```

```
    }  
  } // general-statistics-sent-received  
  
  grouping endpoint-grp {  
    description "A grouping that defines the structure of " +  
      "an endpoint";  
    choice ac-or-pw-or-redundancy-grp {  
      description "A choice of attachment circuit or " +  
        "pseudowire or redundancy group";  
      case ac {  
        description "Attachment circuit(s) as an endpoint";  
        list ac {  
          key "name";  
          leaf name {  
            type string;  
            description "Name of attachment circuit. " +  
              "This field is intended to " +  
              "reference standardized " +  
              "layer-2 definitions.";  
          }  
          leaf igmp-snooping-instance {  
            type igmp-snooping-instance-ref;  
            description "Configure igmp-snooping instance under  
the bridge view";  
          }  
          leaf mld-snooping-instance {  
            type mld-snooping-instance-ref;  
            description "Configure mld-snooping instance under the  
bridge view";  
          }  
        }  
        description "An L2VPN instance's " +  
          "attachment circuit list";  
      }  
    }  
    case pw {  
      description "Pseudowire(s) as an endpoint";  
      list pw {  
        key "name";  
        leaf name {  
          type string;  
          description "Name of Pseudowire.";  
        }  
        leaf igmp-snooping-instance {  
          type igmp-snooping-instance-ref;  
        }  
      }  
    }  
  }  
}
```

```

        description "Configure igmp-snooping instance under
the bridge view";
    }
    leaf mld-snooping-instance {
        type mld-snooping-instance-ref;
        description "Configure mld-snooping instance under the
bridge view";
    }

    description "An L2VPN instance's " +
        "pseudowire(s) list";
}
}
case redundancy-grp {
    description "Redundancy group as an endpoint";
    choice primary {
        mandatory true;
        description "primary options";
        case primary-ac {
            description "primary-ac";
            container primary-ac {
                description "Primary AC";
                leaf name {
                    type string;
                    description "Name of attachment circuit. ";
                }
                leaf igmp-snooping-instance {
                    type igmp-snooping-instance-ref;
                    description "Configure igmp-snooping instance
under the bridge view";
                }
                leaf mld-snooping-instance {
                    type mld-snooping-instance-ref;
                    description "Configure mld-snooping instance
under the bridge view";
                }
            }
        } // primary-ac
    } // primary-ac

    case primary-pw {
        list primary-pw {
            key "name";
            leaf name {
                type string;
                description "Name of Pseudowire.";
            }
        }
    }
}

```

```

        leaf igmp-snooping-instance {
            type igmp-snooping-instance-ref;
            description "Configure igmp-snooping instance
under the bridge view";
        }
        leaf mld-snooping-instance {
            type mld-snooping-instance-ref;
            description "Configure mld-snooping instance
under the bridge view";
        }

        description "primary-pw";
    } //primary-pw
} //primary-pw
}
choice backup {
    description "backup options";
    case backup-ac {
        description "backup-ac";
        container backup-ac {
            description "Backup AC";
            leaf name {
                type string;
                description "Name of attachment circuit. ";
            }
            leaf igmp-snooping-instance {
                type igmp-snooping-instance-ref;
                description "Configure igmp-snooping instance
under the bridge view";
            }
            leaf mld-snooping-instance {
                type mld-snooping-instance-ref;
                description "Configure mld-snooping instance
under the bridge view";
            }
        }
    } // backup-ac
} // backup-ac
case backup-pw {
    description "backup-pw";
    list backup-pw {
        key "name";
        leaf name {
            type string;
            description "Name of Pseudowire.";
        }
        leaf igmp-snooping-instance {

```



```

        type igmp-snooping-instance-ref;
        description "Configure igmp-snooping instance
under the bridge view";
    }
    leaf mld-snooping-instance {
        type mld-snooping-instance-ref;
        description "Configure mld-snooping instance
under the bridge view";
    }

        description "backup-pw";
    } //backup-pw
}
}
}
}

/*
 * igmp-snooping-instance
 */
container igmp-snooping-instances {
    description
        "igmp-snooping-instance list";

    list igmp-snooping-instance {
        key "name";
        description
            "IGMP Snooping instance to configure the igmp-
snooping.";

        leaf name {
            type string;
            description
                "Name of the igmp-snooping-instance to configure the igmp
snooping.";
        }

        leaf id {
            type uint32;
            description
                "It is vlan_id or vpls_id.
When igmp-snooping-instance is applied under bridge view, its
value is 0.";
        }
    }
}

```

```
leaf type {
  type enumeration {
    enum "bridge" {
      description "bridge";
    }
    enum "vpls" {
      description "vpls";
    }
  }
  description "The type indicates bridge or vpls.";
}

uses instance-config-attributes-igmp-snooping {
  if-feature per-instance-config;
}

uses instance-state-attributes-igmp-snooping;
} //igmp-snooping-instance
} //igmp-snooping-instances

/*
 * mld-snooping-instance
 */
container mld-snooping-instances {
  description
    "mld-snooping-instance list";

  list mld-snooping-instance {
    key "name";
    description
      "MLD Snooping instance to configure the mld-snooping.";

    leaf name {
      type string;
      description
        "Name of the mld-snooping-instance to configure the mld
snooping.";
    }

    leaf id {
```

```
    type uint32;
    description
      "It is vlan_id or vpls_id.
      When mld-snooping-instance is applied under bridge view, its
value is 0.";
  }

  leaf type {
    type enumeration {
      enum "bridge" {
        description "bridge";
      }
      enum "vpls" {
        description "vpls";
      }
    }
    description "The type indicates bridge or vpls.";
  }

  uses instance-config-attributes-mld-snooping {
    if-feature per-instance-config;
  }

  uses instance-state-attributes-mld-snooping;
} //mld-snooping-instance
} //mld-snooping-instances

container bridges {
  description
    "Apply igmp-mld-snooping instance in the bridge scenario";

  list bridge {
    key name;

    description
      "bridge list";

    leaf name {
      type name-type;
      description
        "bridge name";
    }
  }
}
```

```

        leaf igmp-snooping-instance {
            type igmp-snooping-instance-ref;
            description "Configure igmp-snooping instance under the
bridge view";
        }
        leaf mld-snooping-instance {
            type mld-snooping-instance-ref;
            description "Configure mld-snooping instance under the
bridge view";
        }
        list component {
            key "name";
            description
                " ";

            leaf name {
                type string;
                description
                    "The name of the Component.";
            }
            container bridge-vlan {
                description "bridge vlan";
                list vlan {
                    key "vid";
                    description
                        " ";

                    leaf vid {
                        type vlan-index-type;
                        description
                            "The VLAN identifier to which this entry
applies.";
                    }
                }
            }
            leaf igmp-snooping-instance {
                type igmp-snooping-instance-ref;
                description "Configure igmp-snooping instance
under the vlan view";
            }
            leaf mld-snooping-instance {
                type mld-snooping-instance-ref;
                description "Configure mld-snooping instance
under the vlan view";
            }
            container interfaces {
                description
                    "Interface configuration parameters.";
            }

```

```
list interface {
    key "name";

    description
        "The list of configured interfaces on the
device.";

    leaf name {
        type string;
        description
            "The name of the interface.";
    }
    leaf igmp-snooping-instance {
        type igmp-snooping-instance-ref;
        description "Configure igmp-snooping
instance under the interface view";
    }
    leaf mld-snooping-instance {
        type mld-snooping-instance-ref;
        description "Configure mld-snooping
instance under the interface view";
    }
}
} // interfaces
} // vlan
} // bridge-vlan
} // component
} // bridge
} // bridges

container l2vpn-instances {
    description "Apply igmp-mld-snooping instance in the vpls
scenario";

    list l2vpn-instance {
        key "name";
        description "An VPLS service instance";

        leaf name {
            type string;
            description "Name of VPLS service instance";
        }
    }
}
```

```
        leaf igmp-snooping-instance {
            type igmp-snooping-instance-ref;
            description "Configure igmp-snooping instance under the
12vpn-instance view";
        }
        leaf mld-snooping-instance {
            type mld-snooping-instance-ref;
            description "Configure mld-snooping instance under the
12vpn-instance view";
        }

        list endpoint {
            key "name";
            description "An endpoint";
            leaf name {
                type string;
                description "endpoint name";
            }
            leaf igmp-snooping-instance {
                type igmp-snooping-instance-ref;
                description "Configure igmp-snooping instance under the
interface view";
            }
            leaf mld-snooping-instance {
                type mld-snooping-instance-ref;
                description "Configure mld-snooping instance under the
interface view";
            }

            uses endpoint-grp;

        } //endpoint
    }

    /*
     * RPCs
     */

    rpc clear-igmp-snooping-groups {
        if-feature rpc-clear-groups;
        description
            "Clears the specified IGMP Snooping cache tables.";

        input {
            leaf id {
                type uint32;
            }
        }
    }
}
```

```
        description
          "VLAN ID, VPLS ID, or EVPN ID";
      }

      leaf group {
        type inet:ipv4-address;
        description
          "Multicast group IPv4 address.
          If it is not specified, all IGMP snooping group tables
are
          cleared.";
      }

      leaf source {
        type inet:ipv4-address;
        description
          "Multicast source IPv4 address.
          If it is not specified, all IGMP snooping source-group
tables are
          cleared.";
      }
    }
  } // rpc clear-igmp-snooping-groups

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

    input {
      leaf id {
        type uint32;
        description
          "VLAN ID, VPLS ID, or EVPN ID";
      }

      leaf group {
        type inet:ipv6-address;
        description
          "Multicast group IPv6 address.
          If it is not specified, all MLD snooping group tables are
          cleared.";
      }

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

```
        "Multicast source IPv6 address.  
        If it is not specified, all MLD snooping source-group  
tables are  
        cleared.";  
    }  
    }  
    } // rpc clear-mld-snooping-groups  
}  
<CODE ENDS>
```

#### 4. Security Considerations

The data model defined does not create any security implications.

#### 5. IANA Considerations

This draft does not request any IANA action.

#### 6. Normative References

- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6021] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6021, October 2010.
- [RFC4541] M. Christensen, K. Kimball, F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, May 2006.
- [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 InternetGroup 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.
- [draft-ietf-pim-igmp-mld-yang-01] X. Liu, F. Guo, M. Sivakumar, P. McAllister, A. Peter, "A YANG data model for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD)", draft-ietf-pim-igmp-mld-yang-01, October 28, 2016.
- [draft-ietf-pim-igmp-mld-yang-03] X. Liu, F. Guo, M. Sivakumar, P. McAllister, A. Peter, "A YANG data model for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD)", draft-ietf-pim-igmp-mld-yang-03, March 13, 2017.
- [draft-dsdt-nmda-guidelines-01] M. Bjorklund, J. Schoenwaelder, P. Shafer, K. Watsen, R. Wilton, "Guidelines for YANG Module Authors (NMDA)", draft-dsdt-nmda-guidelines-01, May 2017
- [draft-bjorklund-netmod-rfc7223bis-00] M. Bjorklund, "A YANG Data Model for Interface Management", draft-bjorklund-netmod-rfc7223bis-00, August 21, 2017
- [draft-bjorklund-netmod-rfc7277bis-00] M. Bjorklund, "A YANG Data Model for IP Management", draft-bjorklund-netmod-rfc7277bis-00, August 21, 2017
- [draft-ietf-netmod-revised-datastores-03] M. Bjorklund, J. Schoenwaelder, P. Shafer, K. Watsen, R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-03, July 3, 2017
- [draft-ietf-bess-evpn-yang-02] P. Brissette, A. Sajassi, H. Shah, Z. Li, H. Chen, K. Tiruveedhula, I. Hussain, J. Rabadan, "Yang Data Model for EVPN", draft-ietf-bess-evpn-yang-02, March 13, 2017

[draft-ietf-bess-l2vpn-yang-06] H. Shah, P. Brissette, I. Chen, I. Hussain, B. Wen, K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", draft-ietf-bess-l2vpn-yang-06.txt, June 30, 2017

#### Authors' Addresses

Hongji Zhao  
Ericsson (China) Communications Company Ltd.  
Ericsson Tower, No. 5 Lize East Street,  
Chaoyang District Beijing 100102, P.R. China

Email: hongji.zhao@ericsson.com

Xufeng Liu  
Jabil  
8281 Greensboro Drive, Suite 200  
McLean VA 22102  
USA

EMail: Xufeng\_Liu@jabil.com

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

Email: liuyisong@huawei.com

Anish Peter  
Individual

EMail: anish.ietf@gmail.com

Mahesh Sivakumar  
Cisco Systems  
510 McCarthy Boulevard  
Milpitas, California  
USA

EMail: [masivaku@cisco.com](mailto:masivaku@cisco.com)

