PIM WG
Internet-Draft
Intended status: Standards Track
Expires: October 17, 2020

X. Liu
Volta Networks
Z. Zhang, Ed.
ZTE Corporation
A. Peter
Individual contributor
M. Sivakumar
Juniper networks
F. Guo
Huawei Technologies
P. McAllister
Metaswitch Networks
April 15, 2020

A YANG Data Model for Multicast Source Discovery Protocol (MSDP) draft-ietf-pim-msdp-yang-18

Abstract

This document defines a YANG data model for the configuration and management of Multicast Source Discovery Protocol (MSDP) Protocol.

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 October 17, 2020.

Copyright Notice

Copyright (c) 2020 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
1.1. Terminology 3
1.2. Conventions Used in This Document 3
1.3. Tree Diagrams 4
1.4. Prefixes in Data Node Names 4
2. Design of the Data Model 4
2.1. Scope of Model 4
2.2. Specification 5
3. Module Structure 5
3.1. MSDP Configuration 7
3.2. MSDP State 8
4. MSDP YANG Model 8
5. Security Considerations 24
6. IANA Considerations 26
7. Contributors 26
8. Acknowledgement 26
9. References 26
9.1. Normative References 26
9.2. Informative References 29
Appendix A. Data Tree Example 29
A.1. The global and peer configuration example 29
A.2. The state example 31
A.3. The actions example 34
Authors' Addresses 34

1. Introduction

[RFC3618] introduces the protocol definition of MSDP. This document defines a YANG data model that can be used to configure and manage the MSDP protocol. The operational state data and statistics can also be retrieved by this model.

This model is designed to be used along with other multicast YANG models such as PIM [I-D.ietf-pim-yang], which are not covered in this document.

1.1. Terminology

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

- action
- augment
- container
- choice
- data model
- data node
- grouping
- identity
- leaf
- list
- module
- uses

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

MSDP: Multicast Source Discovery Protocol [RFC3618].

RP: Rendezvous Point [RFC7761]

RPF: Reverse Path Forwarding [RFC7761]

SA: Source-Active [RFC3618].

1.2. 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.

1.3. Tree Diagrams

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

1.4. 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]
rt	ietf-routing	[RFC8349]
if	ietf-interfaces	[RFC8343]
ip	ietf-ip	[RFC8344]
key-chain	ietf-key-chain	[RFC8177]
rt-types	ietf-routing-types	[RFC8294]
acl	ietf-access-control-list	[RFC8519]

Table 1

2. Design of the Data Model

2.1. Scope of Model

The model covers MSDP [RFC3618].

This model can be used to configure and manage the MSDP protocol. 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 [RFC8639] and [RFC8641] can be implemented 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 protocol. 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. This model can be extended, and it has been structured in a way that such extensions can be conveniently made.

2.2. Specification

The configuration data nodes cover global configuration attributes and per peer configuration attributes. The state data nodes include global, per peer, and source-active information. The container "msdp" is the top level container in this data model. The presence of this container is expected to enable MSDP protocol functionality. No notification is defined in this model.

3. Module Structure

This model imports and augments the ietf-routing YANG model defined in [RFC8349]. Both configuration data nodes and state data nodes of [RFC8349] are augmented.

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].

```
module: ietf-msdp
  augment /rt:routing/rt:control-plane-protocols
            /rt:control-plane-protocol:
    +--rw msdp
       +--rw global
       |  +--rw tcp-connection-source?   if:interface-ref
       |  +--rw default-peer* [peer-addr prefix-policy]
                              {filter-policy}?
       |  |  +--rw peer-addr        -> ../../../peers/peer/address
       |  |  +--rw prefix-policy    -> /acl:acls/acl/name
       |  +--rw originating-rp
       |  |  +--rw interface?   if:interface-ref
       |  +--rw sa-filter
       |  |  +--rw in?    -> /acl:acls/acl/name
       |  |  +--rw out?   -> /acl:acls/acl/name
       |  +--rw sa-limit?                uint32
       |  +--rw ttl-threshold?           uint8
       +--rw peers
       |  +--rw peer* [address]
       |     +--rw address                  inet:ipv4-address
       |     +---x clear-peer
```

```
|     +--rw authentication {peer-authentication}?
|     |  +--rw (authentication-type)?
|     |     +--:(key-chain)
|     |     |  +--rw key-chain?          key-chain:key-chain-ref
|     |     +--:(password)
|     |        +--rw key?                string
|     |        +--rw crypto-algorithm?   identityref
|     +--rw enabled?                 boolean
|     +--rw tcp-connection-source?   if:interface-ref
|     +--rw description?             string
|     +--rw mesh-group?              string
|     +--rw peer-as?                 inet:as-number
                                     {peer-as-verification}?
|     +--rw sa-filter
|     |  +--rw in?    -> /acl:acls/acl/name
|     |  +--rw out?   -> /acl:acls/acl/name
|     +--rw sa-limit?                uint32
|     +--rw timer
|     |  +--rw connect-retry-interval?   uint16
|     |  +--rw holdtime-interval?        uint16
|     |  +--rw keepalive-interval?       uint16
|     +--rw ttl-threshold?           uint8
|     +--ro session-state?           enumeration
|     +--ro elapsed-time?            yang:gauge32
|     +--ro connect-retry-expire?    uint32
|     +--ro hold-expire?             uint16
|     +--ro is-default-peer?         boolean
|     +--ro keepalive-expire?        uint16
|     +--ro reset-count?             yang:zero-based-counter32
|     +--ro statistics
|        +--ro discontinuity-time?   yang:date-and-time
|        +--ro error
|        |  +--ro rpf-failure?   uint32
|        +--ro queue
|        |  +--ro size-in?    uint32
|        |  +--ro size-out?   uint32
|        +--ro received
|        |  +--ro keepalive?      yang:counter64
|        |  +--ro notification?   yang:counter64
|        |  +--ro sa-message?     yang:counter64
|        |  +--ro sa-response?    yang:counter64
|        |  +--ro sa-request?     yang:counter64
|        |  +--ro total?          yang:counter64
|        +--ro sent
|           +--ro keepalive?      yang:counter64
|           +--ro notification?   yang:counter64
|           +--ro sa-message?     yang:counter64
|           +--ro sa-response?    yang:counter64
```

```
|          +--ro sa-request?     yang:counter64
|          +--ro total?          yang:counter64
+---x clear-all-peers
+--ro sa-cache
   +--ro entry* [group source-addr]
   |  +--ro group
   |           rt-types:ipv4-multicast-group-address
   |  +--ro source-addr
   |           rt-types:ipv4-multicast-source-address
   |  +--ro origin-rp* [rp-address]
   |  |  +--ro rp-address       inet:ipv4-address
   |  |  +--ro is-local-rp?     boolean
   |  |  +--ro sa-adv-expire?   uint32
   |  +--ro state-attributes
   |     +--ro up-time?             yang:gauge32
   |     +--ro expire?              yang:gauge32
   |     +--ro holddown-interval?   uint32
   |     +--ro peer-learned-from?   inet:ipv4-address
   |     +--ro rpf-peer?            inet:ipv4-address
   +---x clear
      +---w input
         +---w entry!
         |  +---w group
         |           rt-types:ipv4-multicast-group-address
         |  +---w source-addr?
         |           rt-types:ipv4-multicast-source-address
         +---w peer-address?   inet:ipv4-address
         +---w peer-as?        inet:as-number
```

3.1. MSDP Configuration

MSDP operation requires configuration information that is distributed amongst several peers. Several peers may be configured in a mesh-group. The Source-Active information may be filtered by peers.

The configuration modeling branch is composed of MSDP global and peer configurations. The two parts are the most important parts of MSDP.

Besides the fundamental features of MSDP protocol, several optional features are included in the model. These features help the control of MSDP protocol. The peer features and SA features make the deployment and control easier. The connection parameters can be used to control the TCP connection because MSDP protocol is based on TCP. The authentication features make the protocol more secure. The filter features selectively allow operators to prevent SA information from being forwarded to peers.

3.2. MSDP State

MSDP states are composed of MSDP global state, MSDP peer state, statistics information and SA cache information. The statistics information and SA cache information helps the operator to retrieve the protocol condition.

YANG actions are defined to clear the connection of one specific MSDP peer, clear the connections of all MSDP peers, or clear some or all the SA caches.

4. MSDP YANG Model

This module references [RFC3618], [RFC4271], [RFC5925], [RFC6991], [RFC7761], [RFC8177], [RFC8294], [RFC8343], [RFC8344], [RFC8349], [RFC8519].

```
<CODE BEGINS> file "ietf-msdp@2020-04-15.yang"
module ietf-msdp {

  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-msdp";
  prefix msdp;

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

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

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

  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";
  }

  import ietf-key-chain {
    prefix "key-chain";
    reference "RFC 8177: YANG Data Model for Key Chains";
  }

  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)";
  }

  organization
    "IETF PIM (Protocols for IP Multicast) 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:   Zheng Zhang
               <mailto:zzhang_ietf@hotmail.com>

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

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

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

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

  // RFC Ed.: replace XXXX with actual RFC number and remove
  // this note
```

```
  description
    "The module defines the YANG model definitions for
     Multicast Source Discovery Protocol (MSDP).

     Copyright (c) 2020 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
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC
     itself for full legal notices.

     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 (RFC 2119)
     (RFC 8174) when, and only when, they appear in all
     capitals, as shown here.";

  revision 2020-04-15 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for MSDP.";
  }

  /*
   * Features
   */

  feature filter-policy {
    description
      "Support policy configuration of peer/message filtering.";
    reference
      "RFC 8519: YANG Data Model for Network Access Control
       Lists (ACLs)";
  }

  feature peer-as-verification {
    description
      "Support configuration of peer AS number.";
    reference
```

```
      "RFC 4271: A Border Gateway Protocol 4 (BGP-4)";
  }

  feature peer-authentication {
    description
      "Support configuration of peer authentication.";
    reference
      "RFC 8177: YANG Data Model for Key Chains.";
  }

  /*
   * Identities
   */

  identity msdp {
    base rt:control-plane-protocol;
    description "Identity for the Multicast Source Discovery
                 Protocol (MSDP).";
    reference
      "RFC 3618: Multicast Source Discovery Protocol (MSDP)";
  }

  /*
   * Groupings
   */
  grouping authentication-container {
    description
      "Authentication attributes.";
    container authentication {
      if-feature peer-authentication;
      description
        "A container defining authentication attributes.";
      choice authentication-type {
        case key-chain {
          leaf key-chain {
            type key-chain:key-chain-ref;
            description
              "Reference to a key-chain.";
            reference
              "RFC 8177: YANG Data Model for Key Chains.";
          }
        }
        case password {
          leaf key {
            type string;
            description
              "This leaf specifies the authentication key.";
          }
```

```
          leaf crypto-algorithm {
            type identityref {
              base key-chain:crypto-algorithm;
            }
            must "derived-from-or-self(., 'key-chain:md5')" {
              error-message
                "Only the md5 algorithm can be used for MSDP.";
              description "Check for crypto-algorithm.";
            }
            description
              "Cryptographic algorithm associated with key.
               Only the md5 algorithm can be used for MSDP.
               When 'md5' is specified, MSDP control messages
               are secured by TCP MD5 signatures as described
               in RFC 3618 and RFC 5925. Both peers of a
               connection SHOULD be configured to the same
               algorithm for the connection to be established.
               When this leaf is not configured, unauthenticated
               TCP is used.";
            reference
              "RFC 8177: YANG Data Model for Key Chains.
               RFC 5925: The TCP Authentication Option.";
          }
        }
        description
          "Choice of authentication.";
      }
    }
  } // authentication-container

  grouping tcp-connect-source {
    description
      "Attribute to configure peer TCP connection source.";
    leaf tcp-connection-source {
      type if:interface-ref;
      must "/if:interfaces/if:interface[if:name = current()]/"
        + "ip:ipv4/ip:enabled != 'false'" {
        error-message "The interface must have IPv4 enabled.";
        description
          "The interface must have IPv4 enabled.";
        reference
          "RFC 8343: A YANG Data Model for Interface Management";
      }
      description
        "The interface is to be the source for the TCP
         connection. It is a reference to an entry in the global
         interface list.";
    }
```

```
  } // tcp-connect-source

  grouping global-config-attributes {
    description "Global MSDP configuration.";

    uses tcp-connect-source;

    list default-peer {
      if-feature filter-policy;
      key "peer-addr prefix-policy";

      description
        "The default peer accepts all MSDP SA messages.
         A default peer is needed in topologies where MSDP peers
         do not coexist with BGP peers. The reverse path
         forwarding (RPF) check on SA messages will fail, and no
         SA messages will be accepted. In these cases, you can
         configure the peer as a default peer and bypass RPF checks.";

      leaf peer-addr {
        type leafref {
          path "../../../peers/peer/address";
        }
        mandatory true;
        description
          "Reference to a peer that is in the peer list.";
      }
      leaf prefix-policy {
        type leafref {
          path "/acl:acls/acl:acl/acl:name";
        }
        description
          "If specified, only those SA entries whose RP is
           permitted in the prefix list are allowed;
           if not specified, all SA messages from the default
           peer are accepted.";
        reference
          "RFC 8519: YANG Data Model for Network Access Control
           Lists (ACLs)";
      }
    } // default-peer

    container originating-rp {
      description
        "The container of Originating RP.";
      leaf interface {
        type if:interface-ref;
        must "/if:interfaces/if:interface[if:name = current()]/"
```

```
              + "ip:ipv4/ip:enabled != 'false'" {
          error-message "The interface must have IPv4 enabled.";
          description
            "The interface must have IPv4 enabled.";
          reference
            "RFC 8343: A YANG Data Model for Interface Management";
        }
        description
          "Reference to an entry in the global interface
           list.
           IP address of the interface used in the RP field of
           an SA message entry. When Anycast RPs are used, all
           RPs use the same IP address. This parameter can be
           used to define a unique IP address for the RP of each
           MSDP peer.
           By default, the software uses the RP address of the
           local system.";
      }
    } // originating-rp

    uses sa-filter-container;

    leaf sa-limit {
      type uint32;
      description
        "A limit on the number of SA entries accepted.
         By default, there is no limit.";
    }
    uses ttl-threshold;
  } // global-config-attributes

  grouping peer-config-attributes {
    description "Per peer configuration for MSDP.";

    uses authentication-container;
    leaf enabled {
      type boolean;
      description
        "'true' if peer is enabled;
         'false' if peer is disabled.";
    }
    uses tcp-connect-source;

    leaf description {
      type string;
      description
        "The peer description.";
    }
```

```
    leaf mesh-group {
      type string;
      description
        "The name of mesh-group which this peer belongs to.";
      reference
        "RFC 3618: Multicast Source Discovery Protocol (MSDP),
                   section 10.2.";
    }
    leaf peer-as {
      if-feature peer-as-verification;
      type inet:as-number;
      description
        "Peer's autonomous system number (ASN). Using peer-as to
         do verification can provide more controlled ability.
         The value can be compared with the BGP peer AS. If they
         are different, the SA comes from this peer may be rejected.
         If the AS number is the same as the local AS, then the
         peer is within the same domain; otherwise, this peer is
         external to the domain. Like the definition and usage
         in BGP.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4)";
    }
    uses sa-filter-container;
    leaf sa-limit {
      type uint32;
      description
        "A limit on the number of SA entries accepted from this
         peer. By default, there is no limit.";
    }
    container timer {
      description "Timer attributes.";
      reference
        "RFC 3618: Multicast Source Discovery Protocol (MSDP),
                   section 5.";
      leaf connect-retry-interval {
        type uint16;
        units seconds;
        default 30;
        description "Peer timer for connect-retry.
                     By default, MSDP peers wait 30 seconds after
                     session is reset.";
      }
      leaf holdtime-interval {
        type uint16 {
          range "3..65535";
        }
        units seconds;
```

```
        default 75;
        description "The SA hold down period of this MSDP peer.";
      }
      leaf keepalive-interval {
        type uint16 {
            range "1..65535";
        }
        units seconds;
        must '. < ../holdtime-interval' {
          error-message
            "The keepalive interval must be smaller than the
             hold time interval";
        }
        default 60;
        description "The keepalive timer of this MSDP peer.";
      }
    } // timer
    uses ttl-threshold;
  } // peer-config-attributes

  grouping peer-state-attributes {
    description "Per peer state attributes for MSDP.";

    leaf session-state {
      type enumeration {
        enum disabled  {
          description "Disabled.";
        }
        enum inactive {
          description "Inactive.";
        }
        enum listen {
          description "Listen.";
        }
        enum connecting {
          description "Connecting.";
        }
        enum established {
          description "Established.";
        }
      }
      config false;
      description
        "Peer session state.";
      reference
        "RFC 3618: Multicast Source Discovery Protocol (MSDP),
                   section 11.";
    }
```

```
    leaf elapsed-time {
      type yang:gauge32;
      units seconds;
      config false;
      description "Elapsed time for being in a state.";
    }
    leaf connect-retry-expire {
      type uint32;
      units seconds;
      config false;
      description "Connect retry expire time of peer connection.";
    }
    leaf hold-expire {
      type uint16;
      units seconds;
      config false;
      description "Hold expire time of peer connection.";
    }
    leaf is-default-peer {
      type boolean;
      config false;
      description "'true' if this peer is one of the default peer.";
    }
    leaf keepalive-expire {
      type uint16;
      units seconds;
      config false;
      description "Keepalive expire time of this peer.";
    }
    leaf reset-count {
      type yang:zero-based-counter32;
      config false;
      description "The reset count of this peer.";
    }

    container statistics {
      config false;
      description
        "A container defining statistics attributes.";

      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
          "A grouping defining error statistics attributes.";
        leaf rpf-failure {
          type uint32;
          description "Number of RPF failures.";
        }
      }

      container queue {
        description
          "A container includes queue statistics attributes.";
        leaf size-in {
          type uint32;
          description
            "The number of messages received from the peer
             currently queued.";
        }
        leaf size-out {
          type uint32;
          description
            "The number of messages queued to be sent to the peer.";
        }
      }

      container received {
        description "Received message counters.";
        uses statistics-sent-received;
      }
      container sent {
        description "Sent message counters.";
        uses statistics-sent-received;
      }
    } // statistics
  } // peer-state-attributes

  grouping sa-filter-container {
    description "A container defining SA filters.";
    container sa-filter {
      description
        "Specifies an access control list (ACL) to filter source
         active (SA) messages coming in to or going out of the
         peer.";
      leaf in {
        type leafref {
```

```
          path "/acl:acls/acl:acl/acl:name";
        }
        description
          "Filters incoming SA messages only.
           The value is the name to uniquely identify a
           policy that contains one or more rules used to
           accept or reject MSDP SA messages.
           If the policy is not specified, all MSDP SA messages are
           accepted.";
        reference
          "RFC 8519: YANG Data Model for Network Access Control
           Lists (ACLs)";
      }
      leaf out {
        type leafref {
          path "/acl:acls/acl:acl/acl:name";
        }
        description
          "Filters outgoing SA messages only.
           The value is the name to uniquely identify a
           policy that contains one or more rules used to
           accept or reject MSDP SA messages.
           If the policy is not specified, all MSDP SA messages are
           sent.";
        reference
          "RFC 8519: YANG Data Model for Network Access Control
           Lists (ACLs)";
      }
    } // sa-filter
  } // sa-filter-container

  grouping ttl-threshold {
    description "Attribute to configure TTL threshold.";
    leaf ttl-threshold {
      type uint8 {
        range 1..255;
      }
      description "Maximum number of hops data packets can
                   traverse before being dropped.";
    }
  } // ttl-threshold

  grouping statistics-sent-received {
    description
      "A grouping defining sent and received statistics attributes.";
    leaf keepalive {
      type yang:counter64;
      description
```

```
        "The number of keepalive messages.";
    }
    leaf notification {
      type yang:counter64;
      description
        "The number of notification messages.";
    }
    leaf sa-message {
      type yang:counter64;
      description
        "The number of SA messages.";
    }
    leaf sa-response {
      type yang:counter64;
      description
        "The number of SA response messages.";
    }
    leaf sa-request {
      type yang:counter64;
      description
        "The number of SA request messages.";
    }
    leaf total {
      type yang:counter64;
      description
        "The number of total messages.";
    }
  } // statistics-sent-received

  /*
   * Data nodes
   */
  augment "/rt:routing/rt:control-plane-protocols/"
         + "rt:control-plane-protocol" {
    when "derived-from-or-self(rt:type, 'msdp:msdp')" {
      description
        "This augmentation is only valid for a routing protocol
         instance of MSDP.";
    }
    description
      "MSDP augmentation to routing control-plane protocol
       configuration and state.";
    container msdp {
      description
        "MSDP configuration and operational state data.";

      container global {
        description
```

```
          "Global attributes.";
        uses global-config-attributes;
      }

      container peers {
        description
          "Containing a list of peers.";
        list peer {
          key "address";
          description
            "List of MSDP peers.";
          leaf address {
            type inet:ipv4-address;
            description
              "The address of the peer";
          }
          action clear-peer {
            description
              "Clears the TCP connection to the peer.";
          }
          uses peer-config-attributes;
          uses peer-state-attributes;
        }
      }

      action clear-all-peers {
        description
          "'All peers' TCP connection are cleared.";
      }

      container sa-cache {
        config false;
        description
          "The SA cache information.";
        list entry {
          key "group source-addr";
          description "A list of SA cache entries.";
          leaf group {
            type rt-types:ipv4-multicast-group-address;
            description "The group address of this SA cache.";
          }
          leaf source-addr {
            type rt-types:ipv4-multicast-source-address;
            description "Source IPv4 address.";
          }
          list origin-rp {
            key "rp-address";
            description "Origin RP information.";
```

```
            leaf rp-address {
              type inet:ipv4-address;
              description
                "The RP address. IP address used in the RP field
                 of an SA message entry.";
              }
              leaf is-local-rp {
                type boolean;
                description
                  "'true' if the RP is local;
                   'false' if The RP is not local.";
              }
              leaf sa-adv-expire {
                type uint32;
                units seconds;
                description
                  "The remaining time duration before expiration
                   of the periodic SA advertisement timer on a
                   local RP.";
              }
            }

            container state-attributes {
              description "SA cache state attributes for MSDP.";

              leaf up-time {
                type yang:gauge32;
                units seconds;
                description
                  "Indicates the duration time when this SA entry is
                   created in the cache. MSDP is a periodic protocol,
                   the value can be used to check the state of
                   SA cache.";
              }
              leaf expire {
                type yang:gauge32;
                units seconds;
                description
                  "Indicates the duration time when this SA entry in
                   the cache times out. MSDP is a periodic protocol,
                   the value can be used to check the state of
                   SA cache.";
              }
              leaf holddown-interval {
                type uint32;
                units seconds;
                description
                  "Hold-down timer value for SA forwarding.";
```

```
            reference
              "RFC 3618: Multicast Source Discovery Protocol
                         (MSDP), section 5.3.";
            }
            leaf peer-learned-from {
              type inet:ipv4-address;
              description
                "The address of the peer that we learned this
                 SA from.";
            }
            leaf rpf-peer {
              type inet:ipv4-address;
              description
                "The address is the SA's originating RP.";
            }
          } // state-attributes
        } // entry

        action clear {
          description
            "Clears MSDP source active (SA) cache entries.";
          input {
            container entry {
              presence "If a particular entry is cleared.";
              description
                "The SA cache (S,G) or (*,G) entry to be cleared. If
                 this is not provided, all entries are cleared.";
              leaf group {
                type rt-types:ipv4-multicast-group-address;
                mandatory true;
                description "The group address";
              }
              leaf source-addr {
                type rt-types:ipv4-multicast-source-address;
                description
                  "Address of multicast source to be cleared. If this
                   is not provided then all entries related to the
                   given group are cleared.";
              }
            }
            leaf peer-address {
              type inet:ipv4-address;
              description
                "Peer IP address from which MSDP SA cache entries have
                 been learned. If this is not provided, entries learned
                 from all peers are cleared.";
            }
            leaf peer-as {
```

```
              type inet:as-number;
              description
                "ASN from which MSDP SA cache entries have been learned.
                 If this is not provided, entries learned from all AS's
                 are cleared.";
            }
          }
        } // clear
      } // sa-cache
    } // msdp
  } // 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 NETCONF access control model [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/msdp,

msdp:global

This subtree specifies the configuration for the MSDP attributes at the global level. Modifying the configuration can cause MSDP default peers to be deleted or the connection to be rebuilt, and unexpected filtering of the SA.

msdp:peers

This subtree specifies the configuration for the MSDP attributes at the peer level. Modifying the configuration will allow unexpected MSDP peer establishment and unexpected SA information learning and advertisement.

The key field writability should be controlled strictly. The key misoperation will break the existing MSDP connection, and the associated SA caches will also be deleted.

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 is msdp subtree and key data node and their sensitivity/vulnerability:

/rt:routing/rt:control-plane-protocols/msdp,

Unauthorized access to any data node of the above subtree can disclose the operational state information of MSDP on this device. For example, the peer information disclosure may lead to a forged connection attack, the ACL nodes' uncorrected modification may lead to the filter errors.

The "key" field is also a sensitive readable configuration. Unauthorized reading may lead to the password leaking. Modification will allow the unexpected rebuilding of connected peers.

Authentication configuration is supported via the specification of key-chains [RFC8177] or the direct specification of key and authentication algorithm. Hence, authentication configuration in the "authentication" container inherits the security considerations of [RFC8177]. This includes the considerations with respect to the local storage and handling of authentication keys.

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

/rt:routing/rt:control-plane-protocols/msdp:clear-peer,

/rt:routing/rt:control-plane-protocols/msdp:clear-sa-cache,

Unauthorized access to any of the above action operations can lead to the MSDP peers connection rebuilding or delete SA records on this device.

6. IANA Considerations

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

The IANA is requested to assign one new URI from the IETF XML registry [RFC3688]. Authors are suggesting the following URI:

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

Registrant Contact: The IESG

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

This document also requests one new YANG module name in the YANG Module Names registry [RFC6020] with the following suggestion:

name: ietf-msdp

namespace: urn:ietf:params:xml:ns:yang:ietf-msdp

prefix: msdp

reference: RFC XXXX

7. Contributors

The authors would like to thank Yisong Liu (liuyisong@huawei.com), Benchong Xu (xu.benchong@zte.com.cn), Tanmoy Kundu (tanmoy.kundu@alcatel-lucent.com) for their valuable contributions.

8. Acknowledgement

The authors would like to thank Stig Venaas, Jake Holland for their valuable comments and suggestions.

9. References

9.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3618] Fenner, B., Ed. and D. Meyer, Ed., "Multicast Source Discovery Protocol (MSDP)", RFC 3618, DOI 10.17487/RFC3618, October 2003, <https://www.rfc-editor.org/info/rfc3618>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <https://www.rfc-editor.org/info/rfc5925>.

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

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

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

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

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

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

[RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <https://www.rfc-editor.org/info/rfc8177>.

[RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <https://www.rfc-editor.org/info/rfc8294>.

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

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

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8344] Bjorklund, M., "A YANG Data Model for IP Management", RFC 8344, DOI 10.17487/RFC8344, March 2018, <https://www.rfc-editor.org/info/rfc8344>.

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

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

[RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, DOI 10.17487/RFC8519, March 2019, <https://www.rfc-editor.org/info/rfc8519>.

9.2. Informative References

[I-D.ietf-pim-yang]
Liu, X., McAllister, P., Peter, A., Sivakumar, M., Liu, Y., and f. hu, "A YANG Data Model for Protocol Independent Multicast (PIM)", draft-ietf-pim-yang-17 (work in progress), May 2018.

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

[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>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

Appendix A. Data Tree Example

This section contains an example of an instance data tree in JSON encoding [RFC7951], containing configuration data.

A.1. The global and peer configuration example

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth1",
        "description": "An interface with MSDP enabled.",
        "type": "iana-if-type:ethernetCsmacd",
        "ietf-ip:ipv4": {
          "forwarding": true,
```

```
              "address": [
                {
                  "ip": "192.0.2.1",
                  "prefix-length": 24
                }
              ]
            }
          }
        ]
      },
      "ietf-access-control-list:acls": {
        "acl": [
          {
            "name": "msdp-default-peer-policy",
            "type": "ietf-access-control-list:ipv4-acl-type",
            "aces": {
              "ace": [
                {
                  "name": "accept",
                  "actions": {
                    "forwarding": "ietf-access-control-list:accept"
                  }
                }
              ]
            }
          }
        ]
      },
      "ietf-routing:routing": {
        "router-id": "203.0.113.1",
        "control-plane-protocols": {
          "control-plane-protocol": [
            {
              "type": "ietf-msdp:msdp",
              "name": "msdp-1",
              "ietf-msdp:msdp": {
                "global": {
                  "tcp-connection-source": "eth1",
                  "default-peer": [
                    {
                      "peer-addr": "198.51.100.8",
                      "prefix-policy": "msdp-default-peer-policy"
                    }
                  ],
                  "originating-rp": {
                    "interface": "eth1"
                  },
                  "sa-limit": 0,
```

```
                  "ttl-threshold": 1
                },
                "peers":{
                  "peer":[
                    {
                      "address": "198.51.100.8",
                      "enabled": true,
                      "tcp-connection-source": "eth1",
                      "description": "x",
                      "mesh-group": "x",
                      "peer-as": 100,
                      "sa-limit": 0,
                      "timer":{
                        "connect-retry-interval": 0,
                        "holdtime-interval": 3,
                        "keepalive-interval": 1
                      },
                      "ttl-threshold": 1
                    }
                  ]
                }
              }
            }
          ]
        }
      }
    }
```

A.2. The state example

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth1",
        "description": "An interface with MSDP enabled.",
        "type": "iana-if-type:ethernetCsmacd",
        "phys-address": "00:00:5e:00:53:01",
        "oper-status": "up",
        "statistics": {
          "discontinuity-time": "2020-02-22T11:22:33+02:00"
        },
        "ietf-ip:ipv4": {
          "forwarding": true,
          "mtu": 1500,
          "address": [
            {
              "ip": "192.0.2.1",
```

```
                "prefix-length": 24,
                "origin": "static"
              }
            ]
          }
        }
      ]
    },
    "ietf-access-control-list:acls": {
      "acl": [
        {
          "name": "msdp-default-peer-policy",
          "type": "ietf-access-control-list:ipv4-acl-type",
          "aces": {
            "ace": [
              {
                "name": "accept",
                "actions": {
                  "forwarding": "ietf-access-control-list:accept"
                }
              }
            ]
          }
        }
      ]
    },
    "ietf-routing:routing": {
      "router-id": "203.0.113.1",
      "control-plane-protocols": {
        "control-plane-protocol": [
          {
            "type": "ietf-msdp:msdp",
            "name": "msdp-1",
            "ietf-msdp:msdp":{
              "global":{
                "tcp-connection-source": "eth1",
                "default-peer": [
                  {
                    "peer-addr": "198.51.100.8",
                    "prefix-policy": "msdp-default-peer-policy"
                  }
                ],
                "originating-rp": {
                  "interface": "eth1"
                },
                "sa-limit": 0,
                "ttl-threshold": 1
              },
```

```
                "peers":{
                  "peer":[
                    {
                      "address": "198.51.100.8",
                      "enabled": true,
                      "tcp-connection-source": "eth1",
                      "description": "x",
                      "mesh-group": "x",
                      "peer-as": 100,
                      "sa-limit": 0,
                      "timer":{
                        "connect-retry-interval": 0,
                        "holdtime-interval": 3,
                        "keepalive-interval": 1
                      },
                      "ttl-threshold": 1,
                      "session-state": "established",
                      "elapsed-time": 5,
                      "is-default-peer": true,
                      "keepalive-expire": 1,
                      "reset-count": 1,
                      "statistics": {
                        "discontinuity-time": "2020-02-22T12:22:33+02:00"
                      }
                    }
                  ]
                },
                "sa-cache": {
                  "entry": [
                    {
                      "group": "233.252.0.23",
                      "source-addr": "198.51.100.8",
                      "origin-rp": [
                        {
                          "rp-address": "203.0.113.10",
                          "is-local-rp": false,
                          "sa-adv-expire": 150
                        }
                      ],
                      "state-attributes": {
                        "up-time": 20,
                        "expire": 120,
                        "holddown-interval": 150,
                        "peer-learned-from": "203.0.113.10",
                        "rpf-peer": "203.0.113.10"
                      }
                    }
                  ]
```

```
              }
            }
          }
        ]
      }
    }
  }
```

A.3. The actions example

This example shows the input data (in JSON) for executing an "sa-cache clear" action to clear the cache of all entries which match the group address of 233.252.0.23.

```
  {
    "ietf-msdp:sa-cache":{
      "input":{
        "entry":{
          "group":"233.252.0.23"
        }
      }
    }
  }
```

Authors' Addresses

Xufeng Liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com

Zheng Zhang (editor)
ZTE Corporation
No. 50 Software Ave, Yuhuatai Distinct
Nanjing
China

Email: zzhang_ietf@hotmail.com

Anish Peter
Individual contributor

Email: anish.ietf@gmail.com

Mahesh Sivakumar
Juniper networks
1133 Innovation Way
Sunnyvale, CALIFORNIA 94089
USA

Email: sivakumar.mahesh@gmail.com

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

Email: guofeng@huawei.com

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

Email: pete.mcallister@metaswitch.com