Spring Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 28, 2018

D. Jain
K. Raza
Cisco
B. Decraene
Orange
Z. Jiang
Tencent

Jun 26, 2018

YANG data model for BGP Segment Routing Extensions
draft-dhjain-spring-bgp-sr-yang-00.txt

Abstract

This document defines a YANG data model that can be used to configure and manage Segment Routing extensions in BGP.

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 December 28, 2018.

Copyright Notice

Table of Contents

1. Introduction

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g. ReST) [RFC8040] and encodings other than XML (e.g. JSON)

[RFC7951] are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interfaces, such as CLI and programmatic APIs.

This document defines the YANG model for Segment Routing specific extensions in BGP.

1.1. Requirements Language

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.

2. BGP Segment Routing Yang model

2.1. Overview

Segment Routing (SR), as defined in [I-D.ietf-spring-segment-routing], leverages the source routing paradigm where a node steers a packet through an ordered list of instructions, called segments. SR, thus, allows enforcing a flow through any topological path and/or service chain while maintaining per-flow state only at the ingress nodes to the SR domain.

When applied to ipv6 data-plane (i.e. SRv6), the ordered set of instructions are realized via SRv6 SIDs. The various functions and behaviors corresponding to network programming using SRv6 are specified in [I-D.filsfils-spring-srv6-network-programming].

This document defines Yang model for the Segment Routing extensions applicable for BGP as following:

- o Prefix sid extensions in the context of SR MPLS, as described in [I-D.ietf-idr-bgp-prefix-sid].

- o Egress Peer Engineering (EPE) as described in [I-D.ietf-spring-segment-routing-central-epe].

- o BGP signaled SR Policy as described in [I-D.ietf-idr-segment-routing-te-policy].

- o Automatic Steering as described in [I-D.ietf-spring-segment-routing-policy] and [I-D.ietf-idr-segment-routing-te-policy].

- o SRv6 VPN extensions as described in [I-D.draft-dawra-idr-srv6-vpn].

The Yang extensions proposed in this model augment the base BGP model defined in [I-D.ietf-idr-bgp-model].

Note: Base BGP model does not have a common structure for BGP RIB. The placeholder containers defined in this model can be removed once base BGP model has the BGP RIB structure.

The modeling in this document complies with the Network Management Datastore Architecture (NMDA) [RFC8342]. The operational state data is combined with the associated configuration data in the same hierarchy [I-D.ietf-netmod-rfc6087bis]. When protocol states are retrieved from the NMDA operational state datastore, the returned states cover all "config true" (rw) and "config false" (ro) nodes defined in the schema.

2.2. SR Prefix SID (SR MPLS)

Prefix SID attribute in BGP in the context of SR MPLS, carries the label index and SRGB block information.

- o The configuration to attach the label index is modeled as a new route-policy set action. BGP policy actions from the BGP policy module defined in base BGP yang model [I-D.ietf-idr-bgp-model] are augmented for this purpose.

- o The configuration related to SR Mapping Server in the context of BGP prefix SID, is TBD.

- o Prefix SID attribute received with the BGP route is modeled under BGP AF mode for select address families. This information is applicable per route.

2.3. Egress Peer Engineering

Egress Peer Engineering (EPE) in the context of Segment Routing is described in [I-D.ietf-spring-segment-routing-central-epe]. EPE is enabled in the context of BGP neighbor session. Three different types of EPE SIDs namely, Peer node SID, Peer adjacency SID and Peer set SID correspond to the segments required for source routed inter domain paths. EPE SID(s) for each type above, can be statically configured or dynamically allocated by the node. Further, FRR backup policy and backup SID can be specified per EPE. The configuration and state for the EPE parameters is modeled by augmenting the neighbor container defined in the base BGP model

[I-D.ietf-idr-bgp-model]. The configuration and the applied config state are applicable for neighbor-groups as well.

2.4. SR Policy

Architecture for SR Policies is described in [I-D.ietf-spring-segment-routing-policy]. BGP Signaled SR Policies are described in the [I-D.ietf-idr-segment-routing-te-policy]. Following Yang extensions for SR Policy configuration and state data are applicable:

- o Addition of identies extending the BGP-AFI-SAFI base identity. This is to add two new address families namely IPv4 SR-policy and IPv6 SR-policy, as described in [I-D.ietf-idr-segment-routing-te-policy].

- o BGP Signaled SR Policy candidate paths. These refer to the explicit candidate paths signaled via BGP as SAFI NLRIs, state of which is applicable in the context of BGP speaker process. This is modeled by adding SR Policy address family specific container under generic BGP afi-safi list entry defined in the base BGP model [I-D.ietf-idr-bgp-model].

- o On Demand SR Policy candidate paths. These refer to the dynamic candidate paths as described in [I-D.ietf-spring-segment-routing-policy]. There are two parts to this in the context of BGP. A set of authorized SR Policy colors for on demand policy triggers, and the actual instantiated candidate paths per BGP next-hop. New containers and lists are added under BGP global mode to model this information.

- o SR Policy state in the context of BGP speaker. This represents the state SR Policies (regardless of method of instantiation per candidate path). The SR Policy state is maintained in the context of BGP speaker process to realize the Automatic Steering of overlay routes. Automatic Steering extensions are described in the next section.

Note: The common parameters and datatypes for the SR Policy, currently defined in this model, should be imported from SR Policy Manager model, once available.

2.5. Automatic Steering

Automatic Steering (AS) refers to the ability to forward traffic over a SR Policy on the head-end, as described in [I-D.ietf-spring-segment-routing-policy]. When a BGP route is received with the color extended community and if the color value

matches the color of an authorized SR Policy installed on the head-end, the route is programmed to resolve over SR Policy in forwarding. Automatic Steering information associated with the BGP routes is modeled as state information per route.

TBD: The configuration parameters for Automatic Steering are yet to be added as an augmentation to the BGP route policy model. Such as, extensions for opaque color extended community in BGP policy model, and the Color Only (CO) flags controlling the Automatic Steering behavior as described in [I-D.ietf-idr-segment-routing-te-policy].

2.6. SRv6 SIDs

SRv6 extensions defined here are correspond to the VPN programming via SRv6 as described in [I-D.draft-dawra-idr-srv6-vpn]. SRv6 sid allocation mode is applicable in the context of ipv4 unicast and ipv6 unicast SAFI under VPN context. This is modeled by adding new containers under the respective AFI/SAFIs from the base BGP model [I-D.ietf-idr-bgp-model].

The common data types for SRv6 are imported form [I-D.draft-raza-spring-srv6-yang].

TBD: Base BGP model [I-D.ietf-idr-bgp-model], in its current form is not scoped within the context of a Network Instance. Therefore, the context of a VRF is not fully realized. The extensions done in this model should fall within the scope of a VRF, once the top BGP container is linked under Network Instance.

3. Yang Tree

3.1. SR Prefix Sid (SR MPLS)

```
module: ietf-bgp-sr
augment /rpol:routing-policy/rpol:policy-definitions/rpol:policy-definition/rpol
:statements/rpol:statement/rpol:actions/bgp-pol:bgp-actions:
    +--rw set-label-index?   uint32

augment /bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi/bgp:ipv4-labeled-unicast:
    +--ro routes
       +--ro route* [prefix neighbor add-path-id]
          +--ro prefix                inet:ip-prefix
          +--ro neighbor              inet:ip-address
          +--ro add-path-id           uint32
          +--ro prefix-sid
          |  +--ro label-index?       uint32
          |  +--ro originator-srgb
          |     +--ro srgb-ranges* [srgb-min srgb-max]
          |        +--ro srgb-min    rt-types:mpls-label
          |        +--ro srgb-max    rt-types:mpls-label
  augment /bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi/bgp:ipv6-labeled-unicas
t:
    +--ro routes
       +--ro route* [prefix neighbor add-path-id]
          +--ro prefix                inet:ip-prefix
          +--ro neighbor              inet:ip-address
          +--ro add-path-id           uint32
          +--ro prefix-sid
          |  +--ro label-index?       uint32
          |  +--ro originator-srgb
          |     +--ro srgb-ranges* [srgb-min srgb-max]
          |        +--ro srgb-min    rt-types:mpls-label
          |        +--ro srgb-max    rt-types:mpls-label

 ...
```

3.2. Egress Peer Engineering

Egress Peer Engineering Yang Tree applicable to neighbor and peer-group containers

```
module: ietf-bgp-sr

  augment /bgp:bgp/bgp:neighbors/bgp:neighbor:
    +--rw egress-peer-engineering
       +--rw sid-allocation-type?   enumeration
       +--rw explicit-sid?          sid-type
       +--ro allocated-sid?         sid-type
       +--rw peer-set-name?         string
       +--rw backup
       |  +--ro active?        boolean
       |  +--rw backup-type?   enumeration
       |  +--rw backup-peer?   inet:ip-address
       |  +--rw backup-sid?    sid-type
       +--rw peer-adjacency* [first-hop-ipaddress]
          +--rw first-hop-ipaddress    inet:ip-address
          +--ro first-hop-interface?   string
          +--rw sid-allocation-type?   enumeration
          +--rw explicit-sid?          sid-type
          +--ro allocated-sid?         sid-type
          +--rw backup
             +--ro active?        boolean
             +--rw backup-type?   enumeration
             +--rw backup-peer?   inet:ip-address
             +--rw backup-sid?    sid-type
...
```

3.3. SR Policy

On Demand Nexthop (ODN) policies triggered by BGP

```
     augment /bgp:bgp/bgp:global:
       +--rw segment-routing
          +--rw on-demand-policies
          |  +--ro authorized-colors
          |  |  +--ro colors* [color]
          |  |     +--ro color    uint32
          |  +--ro installed-policies
          |     +--ro sr-policy* [color end-point]
          |        +--ro color        uint32
          |        +--ro end-point    inet:ip-address
          +--ro policy-state
             +--ro sr-policy* [color end-point]
                +--ro color                uint32
                +--ro end-point            inet:ip-address
                +--ro policy-state?        enumeration
                +--ro binding-sid?         sid-type
                +--ro steering-disabled?   empty
                +--ro ref-count?           uint32
```

BGP Signaled Explicit SR Policies under ipv4 and ipv6 SR-Policy SAFI

```
     augment /bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi:
       +--rw ipv4-srpolicy
          +--ro explicit-policies
             +--ro sr-policy* [distinguisher color endpoint]
                +--ro distinguisher           uint32
                +--ro color                   uint32
                +--ro endpoint                inet:ip-address
                +--ro preference?             uint32
                +--ro explicit-binding-sid
                |  +--ro binding-sid?       sid-type
                |  +--ro strict?            boolean
                |  +--ro drop-on-invalid?   boolean
                +--ro usable?                 boolean
                +--ro registered?             boolean

     augment /bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi:
       +--rw ipv6-srpolicy
          +--ro explicit-policies
             +--ro sr-policy* [distinguisher color endpoint]
                +--ro distinguisher           uint32
                +--ro color                   uint32
                +--ro endpoint                inet:ip-address
                +--ro preference?             uint32
                +--ro explicit-binding-sid
                |  +--ro binding-sid?       sid-type
                |  +--ro strict?            boolean
                |  +--ro drop-on-invalid?   boolean
                +--ro usable?                 boolean
                +--ro registered?             boolean
```

3.4. Automatic Steering

Yang Tree for Automatic Steering with example of ipv4-unicast SAFI

```
module: ietf-bgp-sr
  augment /bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi/bgp:ipv4-unicast:
    +--ro routes
       +--ro route* [prefix neighbor add-path-id]
          +--ro prefix                union
          +--ro neighbor              inet:ip-address
          +--ro add-path-id           uint32
          +--ro automatic-steering
          |  +--ro color?         -> /bgp:bgp/global/bgp-sr:segment-routing/poli
cy-state/sr-policy/color
          |  +--ro end-point?     -> /bgp:bgp/global/bgp-sr:segment-routing/poli
cy-state/sr-policy/end-point
          |  +--ro co-flag?       enumeration
          |  +--ro binding-sid?   -> /bgp:bgp/global/bgp-sr:segment-routing/poli
cy-state/sr-policy/binding-sid


 ...
```

3.5. SRv6 SIDs

SRv6 SID allocation mode in the context of select AFs

```
module: ietf-bgp-sr
  augment /bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi/bgp:ipv4-unicast:
    +--rw segment-routing
       +--rw srv6
          +--rw sid-alloc-mode?   enumeration
  augment /bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi/bgp:ipv6-unicast:
    +--rw segment-routing
       +--rw srv6
          +--rw sid-alloc-mode?   enumeration

   ...
```

SRv6 Received and local SID tree

```
 augment /bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi/bgp:l3vpn-ipv4-unicast:
    +--ro routes
       +--ro route* [rd prefix neighbor add-path-id]
          +--ro rd                    rt-types:route-distinguisher
          +--ro prefix                union
          +--ro neighbor              inet:ip-address
          +--ro add-path-id           uint32
          +--ro srv6
             +--ro received-sids* [received-sid]
             |  +--ro received-sid    srv6-types:srv6-sid
             +--ro local-sids* [local-sid]
                +--ro local-sid    srv6-types:srv6-sid
                +--ro locator?     string
  augment /bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi/bgp:l3vpn-ipv6-unicast:
    +--ro routes
       +--ro route* [rd prefix neighbor add-path-id]
          +--ro rd                    rt-types:route-distinguisher
          +--ro prefix                union
          +--ro neighbor              inet:ip-address
          +--ro add-path-id           uint32
          +--ro srv6
             +--ro received-sids* [received-sid]
             |  +--ro received-sid    srv6-types:srv6-sid
             +--ro local-sids* [local-sid]
                +--ro local-sid    srv6-types:srv6-sid
                +--ro locator?     string
```

...

4. Yang Module

```
<CODE BEGINS> file "ietf-bgp-sr@2018-06-26.yang"

module ietf-bgp-sr {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-bgp-sr";
  // replace with IANA namespace when assigned
  prefix bgp-sr ;

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

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

```
  import ietf-routing-policy {
    prefix rpol;
  }

  import ietf-bgp {
    prefix bgp;
  }

  import ietf-bgp-policy {
    prefix bgp-pol;
  }

  import ietf-bgp-types {
    prefix bgp-types;
  }

  import ietf-srv6-types {
    prefix srv6-types;
  }

  organization
     "IETF Spring Working Group";

  contact
     "Spring working group - spring@ietf.org";

  description
    "This YANG module defines a data model to configure and
     manage segment routing extensions in BGP.

     Terms and Acronyms

     AF : Address Family

     BGP (bgp) : Border Gateway Protocol

     EPE : Egress Peer Engineering

     EVPN: Ethernet VPN

     SR : Segment Routing

     SID : Segment Identifier

     SRv6 : Segment Routing with IPv6 Data plane

     VPN : Virtual Private Network
```

```
     VRF : Virtual Routing and Forwarding

    ";

  revision 2018-06-26 {
    description
      "Initial revision" ;
    reference "";
  }

  //
  // New identities and typedefs for SR extensions
  //

  // SR Policy SAFI identities
  identity IPV4_SRPOLICY {
    base bgp-types:AFI_SAFI_TYPE;
    description
     "IPv4 SR Policy (AFI,SAFI = 1,73)";
    reference "TBD";
  }

  identity IPV6_SRPOLICY {
    base bgp-types:AFI_SAFI_TYPE;
    description
      "IPv6 SR Policy (AFI,SAFI = 2,73)";
    reference "TBD";
  }

  // Sid type union
  typedef sid-type {
    type union {
      type rt-types:mpls-label;
      type srv6-types:srv6-sid;
    }
    description "Type definition for Segment Identifier. This is
                 a union type which can be either a SR MPLS SID in the
                 form of a label,  or a SRv6 SID in the form of
                 an IPv6 address.";
    reference "TBD";
  }

  //
  // SR Prefix SID related groupings
  //

  // Prefix SID attribute state in a route
  grouping sr-route-prefix-sid {
```

```
    description "SR Prefix SID attribute associated with BGP Route";
    container prefix-sid {
      description "Prefix SID attribute";
      leaf label-index {
        type uint32;
        description "Label Index TLV carried with Prefix SID";
      }
      container originator-srgb {
        description "SRGB info of the originating node,
                     as signaled in the originator SRGB TLV";
        list srgb-ranges {
          key "srgb-min srgb-max";
          description "Concatenated ranges building the SRGB block";

          leaf srgb-min {
            type rt-types:mpls-label;
            description "Range min";
          }
          leaf srgb-max {
            type rt-types:mpls-label;
            description "Range max";
          }
        }
      }
    }
  }

  //
  // SR Egress Peer Engineering (EPE) related groupings
  //
  grouping epe-sid-alloc-mode {
    description "Common grouping for EPE mode and SID";
    leaf sid-allocation-type {
      type enumeration {
        enum EXPLICIT {
          description "EPE SID is configured";
        }
        enum DYNAMIC {
          description "EPE SID is generated by node";
        }
      }
      default "DYNAMIC";
      description "SID allocation mode specifies whether the EPE SID
                   is explicitly configured value, or a dynamically allocated
                   value by the node. This applicable for EPE peer SID, EPE peer
                   adjacency SID and Peer set SID, depending on the context
                   it is configured.";
    }
```

```
    leaf explicit-sid {
      when "../mode = 'EXPLICIT'";
      type sid-type;
      description "Explicitly configured EPE SID value, when the sid-allocation-
type
                   is EXPLICIT";
    }

    leaf allocated-sid {
      type sid-type;
      config false;
      description "EPE SID value allocated by the node. When the sid allocation
type
                   is DYNAMIC, this would be a SID allocated by the node. In the
 case
                   of EXPLICIT allocation type, this would typically be the expl
icit sid
                   value configured by the user";
    }
  }

  grouping epe-backup-info {
    description "Parameters for EPE backup SID selection";
    container backup {
      description "Backup policy for this EPE";
      leaf active {
        type boolean;
        config false;
        description
          "Boolean indicating if the backup as per requested policy is
           active for this EPE. Typically when EPE Peer, Link or Set is
           down, backup SID as per backup policy, would become active";
      }
      leaf backup-type {
        type enumeration {
          enum PeerNodeSid {
            description "Backup via another Peer Node SID to the
                         same AS. A Peer identifier is also
                         required when this backup-type is
                         selected";
          }
          enum PeerAdjSid {
            description "Backup via remaining Peer Adjacencies to the
                         same peer";
          }
          enum PeerSetSid {
            description "Backup via Remaining PeerNode SIDs in the
                         same PeerSet";
          }
          enum IGP {
            description "Pop the EPE SID and perform IP lookup";
          }
```

```
        }
        default "IGP";
        description "Type of the backup for this EPE";
        reference "TBD";
      }

      leaf backup-peer {
        when "../backup-type = 'PeerNodeSid'";
        type inet:ip-address;
        description "Peer identifier for the case when backup
                     type is PeerNodeSid";
      }

      leaf backup-sid {
        type sid-type;
        description
          "Backup SID (of a EPE Peer, Peer Adjacency or Peer-Set) to be
           used as backup for this EPE";
      }
    }
  }

  grouping epe-config {
    description "Egress Peer Engineering (EPE) config grouping";
    container egress-peer-engineering {
      description "Egress Peer Engineering (EPE) config under BGP Peer";

      uses epe-sid-alloc-mode;

      leaf peer-set-name {
        type string;
        description "Make this EPE peer a member of the named
                     Peer Set.";
      }

      uses epe-backup-info;

      list peer-adjacency {
        key first-hop-ipaddress;
        description "EPE parameters for the adjacency links
                     over which multi-hop peering is setup";

        leaf first-hop-ipaddress {
          type inet:ip-address;
          description "First hop IP address of the link";
        }

        leaf first-hop-interface {
```

```
          type string;
          config false;
          description "The interface corresponding to the link";
        }

        uses epe-sid-alloc-mode;
        uses epe-backup-info;
      }
    }
  }

  //
  // SR Policy Related Groupings
  //
  //Color and Endpoint of the SR Policy
  grouping sr-policy-color-endpoint {
    description "Common grouping for SR Policy Color and
                 Endpoint";
    leaf color {
      type uint32;
      description "Color of the policy";
    }

    leaf end-point {
      type inet:ip-address;
      description "Endpoint of the policy";
    }
  }
  // Authorized colors for On Demand SR Policy programming
  grouping sr-odn-auth-colors {
    description
      "Authorized colors for On Demand (dynamic) SR Policies
       towards BGP nexthops";
    container authorized-colors {
      config false;
      description
        "Authorized colors for On Demand (dynamic) SR policies
         towards BGP nexthops";
      list colors {
        key "color";
        description "List of SR Policy Colors";
        leaf color {
          type uint32;
          description "Color value";
        }
      }
    }
  }
```

```
  grouping sr-policy-cmn-state {
    description "Common state parameters applicable to
                 SR Policies";
    leaf policy-state {
      type enumeration {
        enum UP {
          description "SR Policy state UP";
        }
        enum DOWN {
          description "SR Policy state DOWN";
        }
      }
      description "SR Policy forwarding state";
    }

    leaf binding-sid {
      type sid-type;
      description "Binding SID of the SR Policy";
    }

    leaf steering-disabled {
      type empty;
      description "This attribute is set if steering
                   is disabled on this SR policy";
    }

    leaf ref-count {
      type uint32;
      description "Count of routes steering over this policy";
    }
  }


  //
  // SR Policy State grouping
  //
  grouping sr-policy-state {
    description "SR Policy State";
    container policy-state {
      config false;
      description "SR Policy State";
      list sr-policy {
        key "color end-point";
        description "List of SR Policies";

        uses sr-policy-color-endpoint;

        // State of the SR Policy in BGP
```

```
        uses sr-policy-cmn-state;
      }
    }
  }

  grouping sr-exp-policy-cp-state {
    description "State of BGP signaled SR Policy (explicit)
                 candidate paths";
    container explicit-policies {
      config false;
      description "BGP signaled explicit SR Policies";
      list sr-policy {
        key "distinguisher color end-point";
        description "List of BGP signaled explicit SR Policies";
        leaf distinguisher {
          type uint32;
          description "Distinguisher of the SR Policy
                       candidate path";
        }

        uses sr-policy-color-endpoint;

        leaf preference {
          type uint32;
          description "Preference of the SR Policy candidate path";
        }

        container explicit-binding-sid {
          description "Explicitly supplied Binding SID
                       for this policy";
          leaf binding-sid {
            type sid-type;
            description "Binding SID value";
          }
          leaf strict {
            type boolean;
            description "Boolean indicating that the node
                         must use only the supplied Binding SID
                         for this SR Policy.
                         reference: TBD";
          }
          leaf drop-on-invalid {
            type boolean;
            description "Boolean to indicate drop upon invalid
                         policy, behavior. This overwrites the
                         default behavior of fallback to IGP path
                         , when SR Policy is (or becomes) invalid.
                         reference: TBD";
```

```
          }
        }

        leaf usable {
          type boolean;
          description "Boolean to indicate that the SR Policy is
                       usable on this node.
                       reference: TBD";
        }

        leaf registered {
          type boolean;
          description "Boolean to indicate that the SR policy
                       is registered with policy manager to
                       install the corresponding forwarding entry";

        }

        // TODO: Segment Lists and other parameters from SR Policy model
        //       to be imported here.
      }
    }
  }

  grouping sr-odn-policies {
    description "SR On Demand (dynamic) SR Policies";
    container installed-policies {
      config false;
      description "BGP triggered On Demand (dynamic) SR Policies
                   corresponding to the BGP nexthops";
      list sr-policy {
        key "color end-point";
        description "SR Policy list";
        uses sr-policy-color-endpoint;
      }
    }
  }

  grouping sr-policy-steering-state {
    description "Per route Automatic Steering parameters";
    container automatic-steering {
      description "Per route Automatic Steering parameters";
      leaf color {
        type leafref {
          path "/bgp:bgp/bgp:global/bgp-sr:segment-routing/" +
               "bgp-sr:policy-state/bgp-sr:sr-policy/" +
               "bgp-sr:color";
        }
```

```
        description "Color of the SR Policy being used for
                     Automatic Steering";
      }
      leaf end-point {
        type leafref {
          path "/bgp:bgp/bgp:global/bgp-sr:segment-routing/" +
               "bgp-sr:policy-state/bgp-sr:sr-policy/" +
               "bgp-sr:end-point";
        }
        description "End-point of the SR Policy being used
                     for Automatic Steering";
      }
      leaf co-flag {
        type enumeration {
          enum 00 {
            description "Color-Only flag 00";
          }
          enum 01 {
            description "Color-Only flag 01";
          }
          enum 10 {
            description "Color-Only flag 10";
          }
        }
        default "00";
        description "Color-Only (CO) flags applicable for
                     Automatic Steering of this route";
      }
      leaf binding-sid {
        type leafref {
          path "/bgp:bgp/bgp:global/bgp-sr:segment-routing/" +
               "bgp-sr:policy-state/bgp-sr:sr-policy/" +
               "bgp-sr:binding-sid";
        }
        description "Binding SID of the SR Policy";
      }
    }
  }

  grouping route-key-leafs {
    description "Grouping for key leafs identifying a route";
    leaf prefix {
      type union {
         type inet:ip-prefix;
         type string;
      }
      description "BGP Prefix. This is a temp definition to
                   cover ip-prefix and other NLRI formats.
```

```
                  Import the type once defined in base
                  BGP RIB model";
    }
    leaf neighbor {
      type inet:ip-address;
      description "BGP Neighbor";
    }
    leaf add-path-id {
      type uint32;
      description "Add-path ID";
    }
  }

  grouping common-bgp-route-grouping {
    description "BGP route list" ;
    container routes {
      config false;
      description "BGP Route in local RIB";
      list route {
        key "prefix neighbor add-path-id";
        description "BGP route list";
        uses route-key-leafs;
      }
    }
  }

  grouping common-bgp-vpn-route-grouping {
    description "BGP route list" ;
    container routes {
      config false;
      description "BGP VPN Route in local RIB";
      list route {
        key "rd prefix neighbor add-path-id";
        description "Route List";

        leaf rd {
          type rt-types:route-distinguisher;
          description "Route Distinguisher";
        }
        uses route-key-leafs;
      }
    }
  }

  //
  //  SRv6 extensions related Groupings
  //
```

```
  // SRv6 VPN Sid allocation mode
  grouping srv6-sid-mode {
    description "SRv6 VPN SID allocation mode";
    leaf sid-alloc-mode {
      type enumeration {
        enum per-ce {
          description "Allocate SRv6 SID per CE";
        }
        enum per-route {
          description "Allocate SRv6 SID per prefix";
        }
        enum per-vpn {
          description "Allocate SRv6 SID per VPN";
        }
      }
      description "BGP SRv6 SID allocation model";
    }
  }

  grouping srv6-attr-sid-info {
    description "SRv6 SID info per route";
    container srv6 {
      description "Per Route SRv6 parameters";
      list received-sids {
        key "received-sid";
        description "List of received SRv6 SIDs";
        leaf received-sid {
          type srv6-types:srv6-sid;
          description "Received SID";
        }
      }
      list local-sids {
        key "local-sid";
        description "List of local SRv6 SIDs";
        leaf local-sid {
          type srv6-types:srv6-sid;
          description "Local SID";
        }
        leaf locator {
          type string;
          description "SRv6 Locator";
        }
      }
    }
  }

  //
  // BGP Specific Paramters
```

```
  //
  // Augment AF with route list
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:ipv4-unicast" {
    description
      "Augment BGP SAFI route";
    uses common-bgp-route-grouping;
  }
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:ipv6-unicast" {
    description
      "Augment BGP SAFI route";
    uses common-bgp-route-grouping;
  }
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:ipv4-labeled-unicast" {
    description
      "Augment BGP SAFI route";
    uses common-bgp-route-grouping;

  }
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:ipv6-labeled-unicast" {
    description
      "Augment BGP SAFI route";
    uses common-bgp-route-grouping;
  }
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:l3vpn-ipv4-unicast" {
    description
      "Augment BGP SAFI route";
    uses common-bgp-vpn-route-grouping;
  }
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:l3vpn-ipv6-unicast" {
    description
      "Augment BGP SAFI route";
    uses common-bgp-vpn-route-grouping;
  }
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:l2vpn-evpn" {
    description
      "Augment BGP SAFI route";
    uses common-bgp-vpn-route-grouping;
  }

  // SR Prefix SID Related.
  // Prefix SID label index config via Route Policy
```

```
  augment "/rpol:routing-policy/" +
          "rpol:policy-definitions/rpol:policy-definition/" +
          "rpol:statements/rpol:statement/" +
          "rpol:actions/bgp-pol:bgp-actions" {
    description
       "BGP policy actions to set label index";
    leaf set-label-index {
      type uint32;
      description "Label Index";
    }
  }

  // Prefix SID label in SAFI route
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:ipv4-labeled-unicast/bgp-sr:routes/bgp-sr:route" {
    description
      "Augment BGP AF Table for SR prefix sid Labels info";
    uses sr-route-prefix-sid;
  }
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:ipv6-labeled-unicast/bgp-sr:routes/bgp-sr:route" {
    description
      "Augment BGP AF Table for SR prefix sid Labels info";
    uses sr-route-prefix-sid;
  }

  // TBD: SR Mapping server related parameters.

  // Egress Peer Engineering (EPE) related.
  // EPE config under neighbor
  augment "/bgp:bgp/bgp:neighbors/bgp:neighbor" {
    description
      "Egress Peer Engineering data";
    uses epe-config;
  }
  augment "/bgp:bgp/bgp:peer-groups/bgp:peer-group" {
    description
      "Egress Peer Engineering data";
    uses epe-config;
  }

  // SR Policy Related
  // On Demand authorized colors table
  // SR Policy state data
  augment "/bgp:bgp/bgp:global" {
    description
      "Segment Routing parameters in BGP global model";
    container segment-routing {
```

```
      description "Segment Routing parameters";
      container on-demand-policies {
        description
          "Segment Routing On Demand Nexthop
           (ODN) SR Policies";
        uses sr-odn-auth-colors;
        uses sr-odn-policies;
      }
      uses sr-policy-state;
    }
  }

  // Steering state in overlay BGP routes
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:ipv4-unicast/bgp-sr:routes/bgp-sr:route" {
    description
      "Augment BGP SAFI route with steering info";
    uses sr-policy-steering-state;
  }
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:ipv6-unicast/bgp-sr:routes/bgp-sr:route" {
    description
      "Augment BGP SAFI route with steering info";
    uses sr-policy-steering-state;
  }
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:ipv4-labeled-unicast/bgp-sr:routes/bgp-sr:route" {
    description
      "Augment BGP SAFI route with steering info";
    uses sr-policy-steering-state;
  }
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:ipv6-labeled-unicast/bgp-sr:routes/bgp-sr:route" {
    description
      "Augment BGP SAFI route with steering info";
    uses sr-policy-steering-state;
  }
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:l3vpn-ipv4-unicast/bgp-sr:routes/bgp-sr:route" {
    description
      "Augment BGP SAFI route with steering info";
    uses sr-policy-steering-state;
  }
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:l3vpn-ipv6-unicast/bgp-sr:routes/bgp-sr:route" {
    description
      "Augment BGP SAFI route with steering info";
    uses sr-policy-steering-state;
```

```
  }
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:l2vpn-evpn/bgp-sr:routes/bgp-sr:route" {
    description
      "Augment BGP SAFI route with steering info";
    uses sr-policy-steering-state;
  }

  // BGP Signaled SR Policy explicit candidate paths state
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi" {
    description "Augment IPv4 SR Policy SAFI list entry";
    container ipv4-srpolicy {
      when "../afi-safi-name = 'bgp-types:IPV4_SRPOLICY'" {
       description
         "Include this container for IPv4 SR Policy specific
          configuration";
      }
      description "IPv4 SR Policy specific parameters";
      uses sr-exp-policy-cp-state;
    }
  }

  augment "/bgp:bgp/bgp:global/bgp:afi-safis/bgp:afi-safi" {
    description "Augment IPv6 SR Policy SAFI list entry";
    container ipv6-srpolicy {
      when "../afi-safi-name = 'bgp-types:IPV6_SRPOLICY'" {
        description
          "Include this container for IPv6 SR Policy specific
           configuration";
      }
      description "IPv6 SR Policy specific parameters";
      uses sr-exp-policy-cp-state;
    }
  }

  // SRv6 VPN SID allocation mode configuration.
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:ipv4-unicast" {
    description
      "Augment BGP global IPv4 unicast AF mode
       to add SR specific parameters";
    container segment-routing {
      description "Segment Routing specific parameters";
      container srv6 {
        description "SRv6 specific parameters";
        uses srv6-sid-mode;
      }
    }
```

```
  }

  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:ipv6-unicast" {
    description
      "Augment BGP global IPv6 unicast AF mode
       to add SR specific parameters";
    container segment-routing {
      description "Segment Routing specific parameters";
      container srv6 {
        description "SRv6 specific parameters";
        uses srv6-sid-mode;
      }
    }
  }

  // SRv6 local and remote sids per route.
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:ipv4-unicast/bgp-sr:routes/bgp-sr:route" {
    description
      "Augment AF route with SRv6 SID info";
    uses srv6-attr-sid-info;
  }
  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:ipv6-unicast/bgp-sr:routes/bgp-sr:route" {
    description
      "Augment AF route with SRv6 SID info";
    uses srv6-attr-sid-info;
  }

  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:l3vpn-ipv4-unicast/routes/route" {
    description
      "Augment AF route with SRv6 SID info";
    uses srv6-attr-sid-info;
  }

  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:l3vpn-ipv6-unicast/bgp-sr:routes/bgp-sr:route" {
    description
      "Augment AF route with SRv6 SID info";
    uses srv6-attr-sid-info;
  }

  augment "/bgp:bgp/bgp:global/bgp:afi-safis/" +
          "bgp:afi-safi/bgp:l2vpn-evpn/bgp-sr:routes/bgp-sr:route" {
    description
      "Augment AF route with SRv6 SID info";
```

```
    uses srv6-attr-sid-info;
  }
}

<CODE ENDS>
```

5. IANA Considerations

6. Security Considerations

The transport protocol used for sending the BGP Segment Routing data MUST support authentication and SHOULD support encryption. The data-model by itself does not create any security implications.

This draft does not change any underlying security issues inherent in [I-D.ietf-idr-bgp-model].

7. Acknowledgements

The authors would like to thank TBD for their detail reviews and comments.

8. References

8.1. Normative References

[I-D.dawra-idr-srv6-vpn]
Dawra, G., Filsfils, C., Dukes, D., Brissette, P., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Decraene, B., and S. Matsushima, "BGP Signaling of IPv6-Segment-Routing-based VPN Networks", draft-dawra-idr-srv6-vpn-03 (work in progress), December 2017.

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Li, Z., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Matsushima, S., Lebrun, D., Decraene, B., Peirens, B., Salsano, S., Naik, G., Elmalky, H., Jonnalagadda, P., and M. Sharif, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-04 (work in progress), March 2018.

[I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-13 (work in progress), May 2018.

[I-D.ietf-idr-bgp-model]
Shaikh, A., Shakir, R., Patel, K., Hares, S., D'Souza, K., Bansal, D., Clemm, A., Zhdankin, A., Jethanandani, M., and X. Liu, "BGP Model for Service Provider Networks", draft-ietf-idr-bgp-model-02 (work in progress), July 2016.

[I-D.ietf-idr-bgp-prefix-sid]
Previdi, S., Filsfils, C., Lindem, A., Sreekantiah, A., and H. Gredler, "Segment Routing Prefix SID extensions for BGP", draft-ietf-idr-bgp-prefix-sid-18 (work in progress), March 2018.

[I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Jain, D., Mattes, P., Rosen, E., and S. Lin, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-03 (work in progress), May 2018.

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

[I-D.ietf-rtgwg-ni-model]
Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Network Instances", draft-ietf-rtgwg-ni-model-04 (work in progress), September 2017.

[I-D.ietf-rtgwg-routing-types]
Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Routing Area Common YANG Data Types", draft-ietf-rtgwg-routing-types-17 (work in progress), October 2017.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d., bogdanov@google.com, b., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-01 (work in progress), June 2018.

[I-D.raza-spring-srv6-yang]
Raza, K., Rajamanickam, J., Liu, X., Hu, Z., Hussain, I., Shah, H., daniel.voyer@bell.ca, d., Elmalky, H., Matsushima, S., Horiba, K., and A. Abdelsalam, "YANG Data Model for SRv6 Base and Static", draft-raza-spring-srv6-yang-01 (work in progress), March 2018.

[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>.

[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>.

[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>.

[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>.

8.2. Informative References

[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>.

Authors' Addresses

Dhanendra Jain
Cisco
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: dhjain@cisco.com

Kamran Raza
Cisco
2000 Innovation Drive
Kanata, ON K2K-3E8
CA

Email: skraza@cisco.com

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Zhichun Jiang
Tencent

Email: zcjiang@tencent.com

IDR
Internet-Draft
Updates: 4684 (if approved)
Intended status: Standards Track
Expires: July 18, 2019

J. Heitz
A. Sajassi
Cisco
I. Bagdonas
Equinix

January 14, 2019

BGP Extra Extended Community
draft-heitz-idr-extra-extended-community-02

Abstract

Auto-derived identifiers are used by applications to enable zero configuration features. Such identifiers are often a combination of primitive identifiers and are thus longer. In addition, existing identifiers have grown longer. IP addresses have grown from 4 octets to 16. AS numbers have grown from 2 octets to 4. In order to accommodate such longer identifiers in BGP extended communities, this document defines a new BGP path attribute, the Extra Extended Communities attribute. It is similar to the Extended Community, but is 24 octets long. Communities are mostly used within ASes under a single administration or between neighboring ASes. Limiting the spread of communities beyond their intended reach and polluting the internet at large is complex and error prone. To simplify this, enhanced transitivity options are provided.

Requirements Language

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

Status of This Memo

This Internet-Draft will expire on July 18, 2019.

Copyright Notice

Table of Contents

1. Introduction

A BGP Extended Community attribute is defined that encodes 24 octet communities. It is similar to the Extended Communities attribute defined in [RFC4360], but larger. To simplify the IANA registries, the transitivity of a Extra Extended Community is not part of the IANA registered type. Any type can be encoded with any transitivity. BGP autonomous system (AS) relationships have become more complex. Several contiguous ASes may be under a common administration. A transitivity is defined that allows a XXC to be sent among these ASes only. Some XXCs may be required to be transferred only between neighboring ASes, even though they are under a different administration. A transitivity type to allow this is defined. Up to now, the range of ASes among which a community is distributed is enforced by routing policies. These policies are sometimes executed in the receiving AS, not under the control of the sending AS. The enhanced transitivity options offerend in this document will simplify policies that are used to distribute communities.

2. BGP Extra Extended Community Attribute

The Extra Extended Communities Attribute is a transitive optional BGP attribute, with the Type Code [to be assigned by IANA]. The attribute consists of a set of "Extra Extended Communities" (XXC). The attribute SHOULD contain at least one XXC.

Each XXC is encoded as a 24-octet quantity, as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| T |    Type   |    Sub-Type   |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+          Value                +
|                                                               +
+                                                               +
|                                                               +
+                                                               +
|                                                               +
+                                                               +
|                                                               +
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The fields are as shown below:

T - Transitivity field (2 bits). This is further described below.

Type - 6 bits. IANA will maintain a registry of types. Four types are described in this document.

Sub-type - 8 bits. IANA will maintain a registry of sub-types for each registered type.

Value - The actual information according to the type and sub-type.

Two XXCs are considered equal when all the 24 octets of the XXCs are equal, except the Transitivity field..

3. Transitivity

The transitivity field determines how BGP speakers transfer the XXC across real Autonomous System (AS) boundaries. The XXC is always transitive between Member-ASes in an AS confederation [RFC5065]. The values are:

0 - Transitive: The XXC is transitive across ASes.

1 - Non-transitive: The XXC is not transitive across ASes.

2 - Administration Transitive: The XXC is transitive across ASes under the same administration only.

3 - One-time Transitive: The XXC is transitive across ASes under the same administration and into an AS under the neighboring administration, but not into an AS under a further administration.

To be not transitive means that a sending speaker MUST not send the community. A speaker that receives an XXC across an AS boundary not allowed by the transitivity of that XXC MUST treat the containing UPDATE message as malformed.

A single administration may own a multitude of contiguous ASes. XXCs with transitivity types 2 and 3 are transitive between these ASes. By default, an EBGP neighbor is assumed to be under a different administration. If an EBGP neighbor session is to a speaker in the same administration, then it needs to be configured as such. No Administration identifiers are required. The configuration just needs to specify "Same Administration". There is no new field in the BGP OPEN message to indicate administration membership.

A BGP speaker that receives a XXC with transitivity 3 from a neighbor in an AS under a different administration SHOULD change the transitivity field of the XXC to 2.

The transitivity of an XXC is intended to limit the travel of the XXC in default conditions. It prevents an XXC from traveling far beyond its intended reach if nothing else is done. That means a speaker is free to change the transitivity of an XXC according to local policy to support specific use cases. As an example, a route server as defined in [RFC7947] MAY choose not to change transitivity from 3 to 2. The definition of an XXC type MAY include a specification of the default transitivity for the type.

The Transitivity field is not implicitly associated with the Type and Sub-Type fields the way they are in Extended Communities. The Transitivity field should be set by the originator based upon individual circumstances at the originator. The transitivity is not assigned by IANA.

4. Capability

BGP speakers that do not implement Extra Extended Communities will transfer XXCs even though they may not be transitive across their AS boundaries. To prevent this, a BGP capability as defined in [RFC5492] is required. The length of the capability is 0. A BGP speaker SHOULD NOT send a XXC, the transitivity of which is not 0, to a speaker from which it has not received the Extra Extended Community Capability in its OPEN message. A BGP speaker SHOULD withdraw a route from a neighbor if that neighbor does not advertise this capability and the route contanis an XXC. These rules are intended to prevent XXCs from leaking outside their intended range. There may be cases where such leaking causes no ill effects and the rules can be safely ignored. Such cases are beyond the scope of this document.

A transitive XXC may be sent to a neighbor that has not sent the capability.

5. Constrained Route Distribution

[RFC4684] defines Constrained Route Distribution. That document is updated as follows:

The Extra Constrained Route Distribution SAFI is defined, the NLRI of which is as follows:

```
+-------------------------------+
| AFI              (2 octets)   |
+-------------------------------+
| SAFI             (1 octets)   |
+-------------------------------+
| origin AS        (4 octets)   |
+-------------------------------+
| XXC value       (24 octets)   |
+                               +
|                               |
+-------------------------------+
```

The fields are as shown below:

AFI - The filter specified in this NLRI applies only to prefixes with this AFI.

SAFI - The filter specified in this NLRI applies only to prefixes with this SAFI.

origin AS - Routes that do not have this origin AS will be blocked by the filter.

XXC value - Routes that do not have this XXC will be blocked by the filter.

This works the same way as [RFC4684] with the following differences:

- The maximum prefix size of this NLRI is 248 bits.

- The minimum prefix size of this NLRI is 56 bits except for the default route, which is 0 bits long.

- The filter applies only to prefixes that have the specified AFI/SAFI.

- The filter applies to any XXC values. The filtered XXC does not need to be designated a "Route Target".

Route targets can be expressed as prefixes, where, for instance, a prefix would encompass all route target extended communities assigned by a given Global Administrator. Route Target prefixes can be aggregated. However if done so, then AFI, SAFI, Transitivity, Route Target Type, Sub-Type and the Global Administrator Route Target fields MUST NOT be aggregated.

The address family "Extra Constrained Route Distribution" cannot filter itself.

6. IPv6-Address-Specific Extra Extended Community type

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| T |     0     |    Sub-Type   |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
+                                                               +
|                      Global Administrator                     |
+                                                               +
|                                                               |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                      Local Administrator                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Type field is 0. The Sub-Type is to be assigned by IANA as detailed in the IANA Considerations section.

The Value field consists of 2 sub-fields:

Global Administrator sub-field: 16 octets

This sub-field contains an IPv6 unicast address assigned by one of the Internet registries to the administration of the service using the XXC.

Local Administrator sub-field: 6 octets

The organization identified by the IP address in the Global Administrator sub-field can encode any information in this sub-field. The format and meaning of the value encoded in this sub-field should be defined by the sub-type of the XXC.

7. IPv4-Address-Specific Extra Extended Community type

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| T |     1     |    Sub-Type   |    Global Administrator       :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:  Global Administrator (cont.) |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
+                                                               +
|                      Local Administrator                      |
+                                                               +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Type field is 1. The Sub-Type is to be assigned by IANA for individual functions.

The Value field consists of 2 sub-fields:

Global Administrator sub-field: 4 octets

This sub-field contains an IPv4 unicast address assigned by one of the Internet registries to the administration of the service using the XXC.

Local Administrator sub-field: 18 octets

The organization identified by the IP address in the Global Administrator sub-field can encode any information in this sub-field. The format and meaning of the value encoded in this sub-field should be defined by the sub-type of the XXC.

8. AS-Specific Extra Extended Community type

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| T |     2     |    Sub-Type   |    Global Administrator       :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:  Global Administrator (cont.) |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                      Local Administrator                      |
+                                                               +
|                                                               |
+                                                               +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Type field is 2. The Sub-Type is to be assigned by IANA for individual functions.

The Value field consists of 2 sub-fields:

Global Administrator sub-field: 4 octets

This sub-field contains a 4-octet Autonomous System number assigned by IANA. Note that an ASN that is less than 65536 in value is represented in 4 octets by setting the higher two octets to 0.

Local Administrator sub-field: 18 octets

The organization identified by the Autonomous System number in the Global Administrator sub-field can encode any information in this sub-field. The format and meaning of the value encoded in this sub-field should be defined by the sub-type of the XXC.

9. Route Target Extra Extended Community Sub-Type

For each of the Types of XXC: IPv4-Address-Specific, IPv6-Address-Specific and AS-Specific, the Sub-Type 2 denotes a Route-Target. It has the same use as the Route Target defined in [RFC4360] Sec. 4. The XXC route targets are independent of the RFC4360 Route Targets. There is no way to convert between the two. Either or both may be used in any deployment.

10. EVPN Extra Extended Community type

This is a Extra Extended Community type with a Value field comprising 22 octets.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| T |     6     |    Sub-Type   |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+         Value                 +
|                                                               |
+                                                               +
|                                                               |
+                                                               +
|                                                               |
+                                                               +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Type field is 6. The Sub-Type is to be assigned by IANA for individual functions. Three functions are defined in this document.

11. EVPN Route Target Extra Extended Community sub-types

Auto-derivation of EVPN Route Targets is described in [RFC7432] Sec. 7.10.1. Because of the limited size of Route Targets using Extended Communities, the auto-derivation is limited to using the 12 bit VLAN ID. With the larger size of the RT-XXC, the complete 32 bits of the Ethernet Tag ID can be used.

EVPN XXC route targets have a Type value of 6. The value of the Sub-Type is

1 - AS-Specific: The most significant 4 octets of the Value field are the Autonomous System Number of the AS where this RT is assigned.

2 - IPv4 Address Specific: The most significant 4 octets of the Value field are an IPv4 unicast address assigned by one of the Internet registries to the administration of the service using the XXC.

3 - IPv6 Address Specific: The most significant 16 octets of the Value field are an IPv6 unicast address assigned by one of the Internet registries to the administration of the service using the XXC.

The Ethernet Tag ID is placed into the least significant octets of the Value field. The remaining octets are 0.

12. EVPN ES-Import Route Target Extra Extended Community sub-type

The ES-Import Route Target as specified in [RFC7432] Sec. 7.6 limits the ESI to 6 octets. Thus it cannot be automatically derived for all ESI types. This ES-Import RT-XXC allows the use of the full 10 octets of ESI. The ES-Import Route Target XXC and the ES-Import Route Target extended community are independent. Either or both may be used in any deployment. There is no conversion from one to the other.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| T |     6     |       4       |              GA               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:            GA (Cont.)         |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
+                              ESI                              +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                              Zero                             +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Type field is 6. The Sub-Type is 4. The fields are as follows:

GA - 4 octets. Global Administrator. This is the Autonomous System Number of the AS where this RT is assigned.

ESI - 10 octets. Ethernet Segment Identifier.

Zero - 8 octets filled with 0. Must be ignored by the receiver.

13. EVPN ESI-EVI Route Target Extra Extended Community sub-type

The ESI-EVI Route Target is used in EVPN route types 7 and 8 to filter routes by both ESI and Ethernet Tag ID. More details are in [I-D.ietf-bess-evpn-igmp-mld-proxy].

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| T |     6     |       5       |              GA               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:            GA (Cont.)         |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                                                               |
+                             ESI                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            EVI-RT                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Zero                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Type field is 6. The Sub-Type is 5. The fields are as follows:

GA - 4 octets. Global Administrator. This is the Autonomous System Number of the AS where this RT is assigned.

ESI - 10 octets. Ethernet Segment Identifier.

EVI-RT - 4 octets. The least significant 4 octets of the Local Administrator field of the route target associated with the EVI.

Zero - 4 octets filled with 0. Must be ignored by the receiver.

14. EVPN Overlay Route Target Extra Extended Community sub-type

This EVPN Overlay Route Target Extra Extended Community type is used to filter routes based upon the identifier used in the specified overlay protocol. It works the same way as the RT described in [RFC8365] Sec. 5.1.2.1. It simply provides more room in the fields to allow auto-derivation for more cases. First, it allows a 4 octet AS number. Second, the Service ID is large enough to fit any of the selected IDs.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| T |     6     |       6       |              GA               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:            GA (Cont.)         |A|   Space     |     D-ID      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                                                               +
|                                                               |
+                                                               +
|                           Service-ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Type field is 6. The Sub-Type is 6. The fields are as follows:

GA - 4 octets. Global Administrator. This is the Autonomous System Number of the AS where this RT is assigned.

A - A single bit indicating if this RT is auto-derived

0 : auto-derived

1 : manually derived

Space - 7 bits. The identifier space appropriate to the service. The following spaces are defined:

0 : VID (802.1Q VLAN ID)

1 : VXLAN

2 : NVGRE

3 : I-SID

4 : EVI

5 : dual-VID (QinQ VLAN ID)

D-ID - 1 octet. The default value of domain-id is zero indicating that only a single numbering space exists for a given technology. However, if there is more than one number space for a given technology (e.g., overlapping VXLAN spaces), then each of the number

spaces need to be identified by their corresponding domain-id starting from 1.

Service-ID - 16 octets. VNI, VSID, I-SID, VID or other identifier as appropriate for the service specified in the Space field. If the contained identifier is less than 16 octets long, then it is placed in the least significant octets of the Service-ID field with the higher octets being filled with 0.

15. Duplicate XXC

Two XXCs are considered duplicate if the values of each field except the Transitivity field match.

A BGP speaker SHOULD NOT send a duplicate XXC. However, this is not an error, but merely suboptimal. The duplication of a XXC has no meaning. A receiver of a duplicate XXC SHOULD silently discard the duplicate. The duplication of a XXC cannot be used to compound its effect. For example if one XXC causes the local preference to be incremented by 5, the presence of two of the same XXC will not increment the LP by 10. OTOH, if one XXC increments the LP by 5 and a different XXC increments it by 10, then the combination will cause an increment of 15.

A BGP speaker that receives duplicate XXCs that differ in the Transitivity MAY silently discard the XXC with the more restrictive transitivity.

16. Error Handling

A BGP Extra Extended Communities attribute SHALL be considered malformed if the length of the BGP Extra Extended Communities Attribute value, expressed in octets, is not a multiple of 24. The error SHALL be handled using the approach of "treat-as-withdraw" as described in Section 2 of [RFC7606]. Receipt of a zero length Extra Extended Communities attribute SHALL be treated as "attribute-discard".

The order in which the XXCs appear in the XXC attribute is not significant. It is not an error for a BGP speaker to propagate a set of XXCs in a different order than in which they were received.

If a field in a specific type of XXC is invalid in another setting, it is not necessarily to be considered invalid in a XXC. For example, 0 is an invalid AS number when used in an AS path attribute. That does not make it invalid as an ASN in the AS-Specific XXC. The

behavior and validity of fields in XXCs are to be defined by a specification of the specific type and sub-type of the XXC.

The receipt of an XXC that violates the transitivity rules SHALL be handled using the approach of "treat-as-withdraw".

17. Security Considerations

TBD

18. IANA Considerations

IANA is requested to assign a SAFI for the Extra Constrained Route Distribution address family.

IANA is requested to assign a BGP path attribute value for the Extra Extended Community attribute.

IANA is requested to create and maintain registries as detailed in the following sections. For each registry, the allocation policies as per [RFC8126] are stated for the ranges of values and some values are allocated by this document.

18.1. Registry: BGP Extra Extended Community Types

Range	Allocation Policy
0-31	First Come First Served
32-47	Experimental
48-63	Standards Action

Value	Description	Reference
0	IPv6-Address-Specific	This RFC
1	IPv4-Address-Specific	This RFC
2	AS-Specific	This RFC
6	EVPN	This RFC

18.2. Registry: IPv6-Address-Specific Extra Extended Community Sub-Types

Range	Allocation Policy
0-191	First Come First Served
192-255	IETF Review

Value	Description	Reference
2	Route Target	RFC4360

18.3. Registry: IPv4-Address-Specific Extra Extended Community Sub-Types

Range	Allocation Policy
0-191	First Come First Served
192-255	IETF Review

Value	Description	Reference
2	Route Target	RFC4360

18.4. Registry: AS-Specific Extra Extended Community Sub-Types

Range	Allocation Policy
0-191	First Come First Served
192-255	IETF Review

Value	Description	Reference
2	Route Target	RFC4360

18.5. Registry: EVPN Extra Extended Community Sub-Types

Range	Allocation Policy
0-191	First Come First Served
192-255	IETF Review

Value	Description	Reference
1	EVPN AS-Specific Route Target	This RFC
2	EVPN IPv4-Specific Route Target	This RFC
3	EVPN IPv6-Specific Route Target	This RFC
4	EVPN ES-Import Route Target	This RFC
5	EVPN ESI-EVI Route Target	This RFC
6	EVPN Overlay Route Target	This RFC

19. References

19.1. 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>.

[RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, DOI 10.17487/RFC4360, February 2006, <https://www.rfc-editor.org/info/rfc4360>.

[RFC4684] Marques, P., Bonica, R., Fang, L., Martini, L., Raszuk, R., Patel, K., and J. Guichard, "Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)", RFC 4684, DOI 10.17487/RFC4684, November 2006, <https://www.rfc-editor.org/info/rfc4684>.

[RFC5065] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", RFC 5065, DOI 10.17487/RFC5065, August 2007, <https://www.rfc-editor.org/info/rfc5065>.

[RFC5492] Scudder, J. and R. Chandra, "Capabilities Advertisement with BGP-4", RFC 5492, DOI 10.17487/RFC5492, February 2009, <https://www.rfc-editor.org/info/rfc5492>.

[RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <https://www.rfc-editor.org/info/rfc7432>.

[RFC7606] Chen, E., Ed., Scudder, J., Ed., Mohapatra, P., and K. Patel, "Revised Error Handling for BGP UPDATE Messages", RFC 7606, DOI 10.17487/RFC7606, August 2015, <https://www.rfc-editor.org/info/rfc7606>.

[RFC7947] Jasinska, E., Hilliard, N., Raszuk, R., and N. Bakker, "Internet Exchange BGP Route Server", RFC 7947, DOI 10.17487/RFC7947, September 2016, <https://www.rfc-editor.org/info/rfc7947>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8365] Sajassi, A., Ed., Drake, J., Ed., Bitar, N., Shekhar, R., Uttaro, J., and W. Henderickx, "A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)", RFC 8365, DOI 10.17487/RFC8365, March 2018, <https://www.rfc-editor.org/info/rfc8365>.

19.2. Informative References

[I-D.ietf-bess-evpn-igmp-mld-proxy] Sajassi, A., Thoria, S., Patel, K., Yeung, D., Drake, J., and W. Lin, "IGMP and MLD Proxy for EVPN", draft-ietf-bess-evpn-igmp-mld-proxy-00 (work in progress), March 2017.

Authors' Addresses

Jakob Heitz
Cisco
170 West Tasman Drive
San Jose, CA, CA 95134
USA

Email: jheitz@cisco.com

Ali Sajassi
Cisco
170 West Tasman Drive
San Jose, CA, CA 95134
USA

Email: sajassi@cisco.com

Ignas Bagdonas
Equinix
London
UK

Email: ibagdona.ietf@gmail.com

Interdomain Routing
Internet-Draft
Intended status: Standards Track
Expires: 6 January 2024

M. Jethanandani
Kloud Services
K. Patel
Arrcus
S. Hares
Huawei
J. Haas
Juniper Networks
5 July 2023

YANG Model for Border Gateway Protocol (BGP-4)
draft-ietf-idr-bgp-model-17

Abstract

This document defines a YANG data model for configuring and managing BGP, including protocol, policy, and operational aspects, such as RIB, based on data center, carrier, and content provider operational requirements.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes a YANG 1.1 [RFC7950] data model for the BGP-4 [RFC4271] protocol, including various protocol extensions, policy configuration, as well as defining key operational state data, including a Routing Information Base (RIB). The model is intended to be vendor-neutral, in order to allow operators to manage BGP configuration in heterogeneous environments with routers supplied by multiple vendors. The model is also intended to be readily mapped to existing implementations to facilitate support from as large a set of routing hardware and software vendors as possible. This module does not support previous versions of BGP, and cannot support establishing and maintaining state information of neighbors with previous versions of BGP.

1.1. Goals and approach

The model covers the base BGP features that are deployed across major implementations and the common BGP configurations in use across a number of operator network deployments. In particular, this model attempts to cover BGP features defined in BGP [RFC4271], BGP Communities Attribute [RFC1997], BGP Extended Communities Attributes [RFC4360], IPv6 Extended Communities Attributes [RFC5701], BGP Large Communities Attributes [RFC8092], BGP Route Reflection [RFC4456], Multiprotocol Extensions for BGP-4 [RFC4760], Autonomous System Confederations for BGP [RFC5065], BGP Route Flap Damping [RFC2439], Graceful Restart Mechanism for BGP [RFC4724], BGP Prefix Origin

Validation [RFC6811], and Advertisement of Multiple Paths in BGP [RFC7911]. It attempts to make the model extensible by using identities for well known standard community types, and the use of "raw" string to support new and experimental type definitions.

Along with configuration of base BGP features, this model also addresses policy configuration, by providing "hooks" for applying policies, and also defining BGP-specific policy features. The BGP policy features are intended to be used with the general routing policy model defined in A YANG Data Model for Routing Policy Management [RFC9067].

The model conforms to the NMDA [RFC8342] architecture. It has support for securing BGP sessions using TCP-AO [RFC5925] or TCP-MD5, and for configuring Bidirectional Forward Detection (BFD) [RFC5880] for fast next hop liveliness checking.

For the base BGP features, the focus of the model described in this document is on providing configuration and operational state information relating to:

* The global BGP instance, and neighbors whose configuration is specified individually, or templated with the use of peer-groups.

* The address families that are supported by peers, and the global configuration which relates to them.

* The policy configuration "hooks" and BGP-specific policy features that relate to a neighbor - controlling the import and export of NLRIs.

* BGP RIB contents.

As mentioned earlier, any configuration items that are deemed to be widely available in existing major BGP implementations are included in the model. Additional, more esoteric, configuration items that are not commonly used, or only available from a single implementation, are omitted from the model with an expectation that they will be available in companion modules that augment or extend the current model. This allows clarity in identifying data that is part of the vendor-neutral base model.

Where possible, naming in the model follows conventions used in available standards documents, and otherwise tries to be self-explanatory with sufficient descriptions of the intended behavior. Similarly, configuration data value constraints and default values, where used, are based on recommendations in current standards documentation, or those commonly used in multiple implementations.

Since implementations can vary widely in this respect, this version of the model specifies only a limited set of defaults and ranges with the expectation of being more prescriptive in future versions based on actual operator use.

1.2. Note to RFC Editor

This document uses several placeholder values throughout the document. Please replace them as follows and remove this note before publication.

RFC XXXX, where XXXX is the number assigned to this document at the time of publication.

2023-07-05 with the actual date of the publication of this document.

1.3. 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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.4. Abbreviations

Abbreviation	
AFI	Address Family Identifier
BFD	Bidirectional Forward Detection
NLRI	Network Layer Reachability Information
NMDA	Network Management Datastore Architecture
RIB	Routing Information Base
SAFI	Subsequent Address Family Identifier
VRF	Virtual Routing and Forwarding

Table 1

2. Model overview

The BGP model is defined across several YANG modules and submodules, but at a high level is organized into six elements:

* base protocol configuration -- configuration affecting BGP protocol-related operations, defined at various levels of hierarchy.

* multi-protocol configuration -- configuration affecting individual address-families within BGP Multiprotocol Extensions for BGP-4 [RFC4760].

* neighbor configuration -- configuration affecting an individual neighbor within BGP.

* neighbor multi-protocol configuration -- configuration affecting individual address-families for a neighbor within BGP.

* policy configuration -- hooks for application of the policies defined in A YANG Data Model for Routing Policy Management [RFC9067] that act on routes sent (received) to (from) peers or other routing protocols and BGP-specific policy features.

* operational state -- variables used for monitoring and management of BGP operations.

These modules also make use of standard Internet types, such as IP addresses and prefixes, autonomous system numbers, etc., defined in Common YANG Data Types [RFC6991].

2.1. BGP protocol configuration

The BGP protocol configuration model is organized hierarchically, much like the majority of router implementations. That is, configuration items can be specified at multiple levels, as shown below.

```
module: ietf-bgp

  augment /rt:routing/rt:control-plane-protocols
            /rt:control-plane-protocol:
    +--rw bgp
       +--rw global!
       |  +--rw as                        inet:as-number
       |  +--rw identifier?               yang:dotted-quad
       |  +--rw distance
       |  |     ...
```

```
          |  +--rw confederation
          |  |     ...
          |  +--rw graceful-restart {bt:graceful-restart}?
          |  |     ...
          |  +--rw use-multiple-paths
          |  |     ...
          |  +--rw route-selection-options
          |  |     ...
          |  +--rw afi-safis
          |  |     ...
          |  +--rw apply-policy
          |  |     ...
          |  +--ro statistics
          |        ...
          +--rw neighbors
          |  +--rw neighbor* [remote-address]
          |  |     ...
          |  +---n established
          |  |     ...
          |  +---n backward-transition
          |  |     ...
          |  +---x clear {bt:clear-neighbors}?
          |        ...
          +--rw peer-groups
          |  +--rw peer-group* [name]
          |        ...
          +--ro rib
             +--ro attr-sets
             |     ...
             +--ro communities
             |     ...
             +--ro ext-communities
             |     ...
             +--ro ipv6-ext-communities
             |     ...
             +--ro large-communities
             |     ...
             +--ro afi-safis
                   ...
```

Users may specify configuration at a higher level and have it apply to all lower-level items, or provide overriding configuration at a lower level of the hierarchy. Overriding configuration items are optional, with neighbor-specific configuration being the most specific or lowest level, followed by peer-group, and finally global. Global configuration options reflect a subset of the peer-group or neighbor-specific configuration options which are relevant to the entire BGP instance.

The model makes the simplifying assumption that most of the configuration items are available at all levels of the hierarchy. That is, very little configuration is specific to a particular level in the hierarchy, other than obvious items such as "group-name" only being available for the peer group-level config. A notable exception is for sub-address family configuration where some items are only applicable for a given AFI-SAFI combination.

In order to allow common configuration to be applied to a set of neighbors, all neighbor configuration options are available within a peer-group. A neighbor is associated with a particular peer-group through the use of a peer-group leaf (which provides a reference to a configured item in the peer-group list).

Address-family configuration is made available in multiple points within the model - primarily within the global container, where instance-wide configuration can be set (for example, global protocol parameters, the BGP best-path route selection options, or global policies relating to the address-family); and on a per-neighbor or per-peer-group basis, where address-families can be enabled or disabled, and policy associated with the parent entity applied. Within the afi-safi container, generic configuration that applies to all address-families (e.g., whether the AFI-SAFI is enabled) is presented at the top-level, with address-family specific containers made available for options relating to only that AFI-SAFI. Within the current revision of the model a generic set of address-families, and common configuration and state options are included - further work is expected to add additional parameters to this area of the model.

The model supports ipv4-unicast and ipv6-unicast address-families and defers the remaining AFI/SAFI to other or future drafts:

```
+--rw bgp
  +--rw global!
     +--rw afi-safis
        +--rw afi-safi* [afi-safi-name]
           +--rw afi-safi-name              identityref
           |
           +--rw ipv4-unicast
           |     ...
           +--rw ipv6-unicast
           |     ...
           +--rw ipv4-labeled-unicast
           |     ...
           +--rw ipv6-labeled-unicast
           |     ...
           +--rw l3vpn-ipv4-unicast
           |     ...
           +--rw l3vpn-ipv6-unicast
           |     ...
           +--rw l3vpn-ipv4-multicast
           |     ...
           +--rw l3vpn-ipv6-multicast
           |     ...
           +--rw l2vpn-vpls
           |     ...
           +--rw l2vpn-evpn
           |     ...
```

2.2. Policy configuration overview

The BGP policy configuration model augments the generic YANG routing policy model described in A YANG Data Model for Routing Policy Management [RFC9067], which represents a condition-action policy framework for routing. This model adds BGP-specific conditions (e.g., matching on the community attribute), and actions (e.g., setting local preference) to the generic policy framework.

Policies that are defined in the routing-policy model are referenced in multiple places within the model:

* within the global instance, where a policy applies to all address-families for all peers.

* on a global AFI-SAFI basis, where policies apply to all peers for a particular address-family.

* on a per-peer-group or per-neighbor basis - where the policy applies to all address-families for the particular group or neighbor.

* on a per-afi-safi basis within a neighbor or peer-group context, where the policy is specific to the AFI-SAFI for a a specific neighbor or group.

```
module: ietf-bgp-policy

  augment /rt-pol:routing-policy/rt-pol:defined-sets:
    +--rw bgp-defined-sets
          ...
  augment /rt-pol:routing-policy/rt-pol:policy-definitions
            /rt-pol:policy-definition/rt-pol:statements
            /rt-pol:statement/rt-pol:conditions:
    +--rw bgp-conditions
          ...
  augment /rt-pol:routing-policy/rt-pol:policy-definitions
            /rt-pol:policy-definition/rt-pol:statements
            /rt-pol:statement/rt-pol:actions:
    +--rw bgp-actions
          ...
```

2.3. BGP RIB overview

The RIB data model represents the BGP RIB contents. The model supports five logical RIBs per address family.

An abridged version of the tree shows the RIB portion of the tree diagram.

```
module: ietf-bgp

  augment /rt:routing/rt:control-plane-protocols
            /rt:control-plane-protocol:
    +--rw bgp
       +--ro rib
          +--ro afi-safis
             +--ro afi-safi* [name]
                +--ro name            identityref
                +--ro ipv4-unicast
                |  +--ro loc-rib
                |  |  +--ro routes
                |  |     +--ro route* [prefix origin path-id]
                |  |           ...
                |  +--ro neighbors
                |     +--ro neighbor* [neighbor-address]
                |        +--ro neighbor-address    inet:ip-address
                |        +--ro adj-rib-in-pre
                |        |     ...
                |        +--ro adj-rib-in-post
                |        |     ...
                |        +--ro adj-rib-out-pre
                |        |     ...
                |        +--ro adj-rib-out-post
                |              ...
                +--ro ipv6-unicast
                   +--ro loc-rib
                   |  +--ro routes
                   |     +--ro route* [prefix origin path-id]
                   |           ...
                   +--ro neighbors
                      +--ro neighbor* [neighbor-address]
                         +--ro neighbor-address    inet:ip-address
                         +--ro adj-rib-in-pre
                         |     ...
                         +--ro adj-rib-in-post
                         |     ...
                         +--ro adj-rib-out-pre
                         |     ...
                         +--ro adj-rib-out-post
                               ...
```

2.3.1. Local Routing

The loc-rib is the main BGP routing table for the local routing instance, containing best-path selections for each prefix. The key for the loc-rib will correspond either to a route in the adj-rib-in-post table, described below, or a route redistributed from another protocol.

2.3.2. Pre updates per-neighbor

The adj-rib-in-pre table is a per-neighbor table containing the NLRI updates received from the neighbor before any local input policy rules or filters have been applied. This can be considered the 'raw' updates from a given neighbor.

2.3.3. Post updates per-neighbor

The adj-rib-in-post table is a per-neighbor table containing the routes received from the neighbor that are eligible for best-path selection after local input policy rules have been applied.

The "best-path" leaf is attached to the route selected by the BGP Decision Process Section 9.1 of [RFC4271]. Such routes may be present in the loc-rib table, described above, when local configuration determines that the BGP best-path will be used for that destination.

2.3.4. Pre route advertisements per-neighbor

The adj-rib-out-pre table is a per-neighbor table containing routes eligible for sending (advertising) to the neighbor before output policy rules have been applied.

2.3.5. Post route advertisements per-neighbor

The adj-rib-out-post table is a per-neighbor table containing routes eligible for sending (advertising) to the neighbor after output policy rules have been applied.

3. Relation to other YANG data models

The BGP model augments the Routing Management model A YANG Data Model for Routing Management [RFC8349] which defines the notion of routing, routing protocols, and RIBs. The notion of Virtual Routing and Forwarding (VRF) is derived by using the YANG Schema Mount [RFC8528] to mount the Routing Management module under the YANG Data Model for Network Instances [RFC8529].

Similarly, it augments the A YANG Data Model for Routing Policy Management [RFC9067] to add in BGP-specific conditions and actions to the generic policy model.

4. 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 (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. Some of the subtrees and data nodes and their sensitivity/vulnerability are described here.

* The attribute 'as'. If a user is allowed to change this attribute, it will have the net effect of bringing down the entire routing instance, causing it to delete all the current routing entries, and learning new ones.

* The attribute 'identifier'. If a user is allowed to change this attribute, it will have the net effect of this routing instance re-advertising all its routes.

* The attribute 'distance'. If a user is allowed to change this attribute, it will cause the preference for routes, e.g. external vs internal to change.

* The attribute 'enabled' in the 'confederation' container. This attribute defines whether a local-AS is part of a BGP confederation.

* Finally, there are a whole set of route selection options such as 'always-compare-med', 'ignore-as-path-length' that affect the way the system picks up a particular route. Being able to change will adversely affect how the route selection happens.

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. Some of the subtrees and data nodes and their sensitivity/vulnerability are:

* The list of neighbors, and their attributes. Allowing a user to read these attributes, in particular the address/port information may allow a malicious user to launch an attack at the particular address/port.

* The 'rib' container. This container contains sensitive information such as attribute sets, communities, extended communities, IPv6 extended communities, and large communities. Being able to read the contents of this container will allow a malicious user to understand how the system decide how to route a packet, and thus try to affect a change.

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:

* The model allows for routes to be cleared using the 'clear' RPC operations, causing the entire RIB table to be cleared.

* The model allows for statistics to be cleared by the 'clear' RPC operation, causing all the individual statistics to be cleared.

* The model also allows for neighbors that have been learnt by the system to be cleared by using the 'clear' RPC operation.

BGP OPSEC [RFC7454] describes several policies that can be used to secure BGP. In particular, it recommends securing the underlying TCP session and to use the Generalized TTL Security Mechanism (GTSM) [RFC5082] capability to make it harder to spoof a BGP session. This module allows implementations that want to support the capability to configure a TTL value, under a feature flag. It also defines a container 'secure-session' that contains a choice of using TCP-Authentication Option (TCP-AO) [RFC5925], or Protection of BGP Sessions via the TCP MD5 Signature Option [RFC2385]. The support for MD5 has been made available for legacy reasons. However, because of vulnerabilities in MD5 as described in MD5 and HMAC-MD5 Security Considerations [RFC6151], operators are strongly encouraged to use TCP-AO or other methods to secure their BGP session.

5. IANA Considerations

This document registers six URIs and six YANG modules.

5.1. URI Registration

Following the format in the IETF XML registry [RFC3688] [RFC3688], the following registration is requested to be made:

```
URI:urn:ietf:params:xml:ns:yang:ietf-bgp
URI:urn:ietf:params:xml:ns:yang:ietf-bgp-policy
URI:urn:ietf:params:xml:ns:yang:iana-bgp-community-types
URI:urn:ietf:params:xml:ns:yang:iana-bgp-notification
URI:urn:ietf:params:xml:ns:yang:iana-bgp-rib-types
URI:urn:ietf:params:xml:ns:yang:iana-bgp-types
```

Registrant Contact: The IESG. XML: N/A, the requested URI is an XML namespace.

5.2. YANG Module Name Registration

This document registers six YANG modules in the YANG Module Names registry YANG [RFC6020].

```
name: ietf-bgp
namespace: urn:ietf:params:xml:ns:yang:ietf-bgp
prefix: bgp
reference: RFC XXXX

name: ietf-bgp-policy
namespace: urn:ietf:params:xml:ns:yang:ietf-bgp-policy
prefix: bp
reference: RFC XXXX

name: iana-bgp-community-types
namespace: urn:ietf:params:xml:ns:yang:iana-bgp-community-types
prefix: bct
reference: RFC XXXX

name: iana-bgp-notification
namespace: urn:ietf:params:xml:ns:yang:iana-bgp-notification
prefix: bn
reference: RFC XXXX

name: iana-bgp-types
namespace: urn:ietf:params:xml:ns:yang:iana-bgp-types
prefix: bt
reference: RFC XXXX

name: iana-bgp-rib-types
namespace: urn:ietf:params:xml:ns:yang:iana-bgp-rib-types
prefix: brt
reference: RFC XXXX
```

6. YANG modules

The modules comprising the BGP configuration and operational model are described by the YANG modules and submodules in the sections below.

The main module, ietf-bgp.yang, includes the following submodules:

* ietf-bgp-capabilities - defines the groupings that are common to the BGP Capabilities feature.

* ietf-bgp-common - defines the groupings that are common across more than one context (where contexts are neighbor, group, global).

* ietf-bgp-common-multiprotocol - defines the groupings that are common across more than one context, and relate to multi-protocol BGP.

* ietf-bgp-common-structure - defines groupings that are shared by multiple contexts, but are used only to create structural elements; i.e., containers (leaf nodes are defined in separate groupings).

* ietf-bgp-neighbor - groupings with data specific to the neighbor context.

* ietf-bgp-rib - grouping for representing BGP RIBs.

* ietf-bgp-rib-attributes - common data definitions for BGP attributes used in BGP RIB tables.

* ietf-bgp-rib-tables - structural data definitions for BGP routing tables.

Additionally, the above modules include the following modules:

* iana-bgp-community-types - common type and identity definitions for BGP Communities.

* ietf-bgp-policy - BGP-specific policy data definitions for use with [RFC9067] (described in more detail Section 2.2).

* iana-bgp-types - common type and identity definitions for BGP, including BGP policy.

* iana-bgp-rib-types - common type and identity definitions for BGP RIB.

7. Structure of the YANG modules

The YANG model can be subdivided between the main module for base items, types, and policy module. It references:

* BGP Communities Attribute [RFC1997]

* Route Refresh Capability for BGP-4 [RFC2918]

* NOPEER Community for BGP [RFC3765]

* BGP Extended Communities Attributes [RFC4360]

* BGP/MPLS IP Virtual Private Networks (VPNs) [RFC4364]

* BGP MED Considerations [RFC4451]

* BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN [RFC4659]

* Graceful Restart Mechanism for BGP [RFC4724]

* Multiprotocol Extentions for BGP-4 [RFC4760]

* Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling [RFC4761]

* Autonomous System Configuration for BGP [RFC5065]

* The Generalized TTL Security Mechanism (GTSM) [RFC5082]

* IPv6 Address Specific BGP Extended Community Attribute [RFC5701]

* Bidirectional Forward Detection (BFD) [RFC5880]

* Bidirectional Forward Detection for IPv4 and IPv6 (Single Hop) [RFC5881]

* Bidirectional Forwarding Detection (BFD) for Multihop Paths [RFC5883]

* The TCP Authentication Option [RFC5925]

* BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs [RFC6514]

* BGP Support for Four-Octet Autonomous System (AS) Number Space [RFC6793]

* Advertisement of Multiple Paths in BGP [RFC7911]

* BGP Large Communities Attributes [RFC8092]

* YANG Key Chain [RFC8177]

* Carrying Label Information in BGP-4 [RFC8277]

* A YANG Data Model for Routing Policy [RFC9067]

* YANG Data Model for Bidirectional Forward Detection [RFC9314]

* Transmission Control Protocol [RFC9293]

* YANG Model for Transmission Control Protocol (TCP) Configuration [I-D.ietf-tcpm-yang-tcp]

7.1. BGP module and submodules

7.1.1. BGP module

```
<CODE BEGINS> file "ietf-bgp@2023-07-05.yang"
module ietf-bgp {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-bgp";
  prefix bgp;

  /*
   * Import and Include
   */

  import ietf-routing {
    prefix rt;
    reference
      "RFC 8349: A YANG Data Model for Routing Management
       (NMDA Version).";
  }
  import ietf-routing-policy {
    prefix rt-pol;
    reference
      "RFC 9067: A YANG Data Model for Routing Policy.";
  }
  import iana-bgp-notification {
    prefix bn;
    reference
      "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
  }
  import iana-bgp-types {
    prefix bt;
    reference
      "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
  }
  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.";
  }
```

```
     include ietf-bgp-capabilities {
       revision-date 2023-07-05;
     }
     include ietf-bgp-common {
       revision-date 2023-07-05;
     }
     include ietf-bgp-common-multiprotocol {
       revision-date 2023-07-05;
     }
     include ietf-bgp-common-structure {
       revision-date 2023-07-05;
     }
     include ietf-bgp-neighbor {
       revision-date 2023-07-05;
     }
     include ietf-bgp-rib {
       revision-date 2023-07-05;
     }
     include ietf-bgp-rib-attributes {
       revision-date 2023-07-05;
     }
     include ietf-bgp-rib-tables {
       revision-date 2023-07-05;
     }

     organization
       "IETF IDR Working Group";
     contact
       "WG Web:   <http://datatracker.ietf.org/wg/idr>
        WG List:  <idr@ietf.org>

        Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                 Keyur Patel (keyur at arrcus.com),
                 Susan Hares (shares at ndzh.com),
                 Jeffrey Haas (jhaas at juniper.net).";

     description
       "This module describes a YANG model for BGP protocol
        configuration. It is a limited subset of all of the
        configuration parameters available in the variety of vendor
        implementations, hence it is expected that it would be augmented
        with vendor-specific configuration data as needed. Additional
        modules or submodules to handle other aspects of BGP
        configuration, including policy, VRFs, VPNs, and additional
        address families are also expected.

        This model supports the following BGP configuration level
        hierarchy:
```

```
         BGP
           |
           +-> [ global BGP configuration ]
             +-> AFI / SAFI global
           +-> peer group
             +-> [ peer group config ]
             +-> AFI / SAFI [ per-AFI overrides ]
           +-> neighbor
             +-> [ neighbor config ]
             +-> [ optional pointer to peer-group ]
             +-> AFI / SAFI [ per-AFI overrides ]
```


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 2023-07-05 {
       description
         "Initial Version";
       reference
         "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
     }

     /*
      * Identity
      */

     identity bgp {
       base rt:routing-protocol;
       description
         "BGP protocol.";
     }
```

```
     /*
      * Groupings
      */

     grouping bgp-errors-common {
       description
         "BGP NOTIFICATION state that is common in the sent and received
          direction.";

       leaf last-notification {
         type yang:date-and-time;
         description
           "The timestamp indicates the time that a BGP
            NOTIFICATION message was last handled.";
         reference
           "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
            4.5.";
       }
       leaf last-error {
         type identityref {
           base bn:bgp-notification;
         }
         description
           "The last NOTIFICATION error for the peer on this
            connection.  If no identity is registered for the
            Error code / Error subcode, this leaf contains the most
            applicable identity for the BGP NOTIFICATION base code.

            The last-error-code and last-error-subcode will always have
            the information received in the NOTIFICATION PDU.";
         reference
           "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
            4.5.";
       }
       leaf last-error-code {
         type uint8;
         description
           "The last NOTIFICATION Error code for the peer on
            this connection.";
         reference
           "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
            4.5.";
       }
       leaf last-error-subcode {
         type uint8;
         description
           "The last NOTIFICATION Error subcode for the peer on
            this connection.";
```

```
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
           4.5.";
      }
    }
    grouping bgp-errors-common-data {
      description
        "BGP NOTIFICATION data state that is common in the sent and
         received direction.";

      leaf last-error-data {
        type binary {
          length "0..65535";
        }
        description
          "Content of the BGP NOTIFICATION PDU's Data field.  This data
           is NOTIFICATION Error code / Error subcode specific.";
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
           4.5.";
      }
    }

    grouping bgp-encapsulated-errors-common {
      description
        "BGP NOTIFICATION state that is common in the sent and received
         direction that has been encapsulated in a CEASE/HARD RESET
         NOTIFICATION.

         Note that these leaves are only present when carrying the RFC
         8538 encapsulated NOTIFICATION state. In this case,
         the last-error-data leaf will carry the encapsulated Data.";

      leaf last-encapsulated-error {
        type identityref {
          base bn:bgp-notification;
        }
        description
          "The last RFC 8538 encapsulated NOTIFICATION error for the
           peer on this connection.  If no identity is registered for
           the Error code / Error subcode, this leaf contains the most
           applicable identity for the BGP NOTIFICATION base code.

           The last-encapsulated-error-code and
           last-encapsulated-error-subcode will always have the
           encapsulated information received in the CEASE/HARD RESET
           NOTIFICATION PDU's encapsulated ErrCode and Subcode
           fields.";
```

```
        reference
          "RFC 8538: Notification Message Support for BGP Graceful
           Restart, Section 3.1.";
      }
      leaf last-encapsulated-error-code {
        type uint8;
        description
          "The last RFC 8538 encapsulated NOTIFICATION Error code for
           the peer on this connection.";
        reference
          "RFC 8538: Notification Message Support for BGP Graceful
           Restart, Section 3.1.";
      }
      leaf last-encapsulated-error-subcode {
        type uint8;
        description
          "The last RFC 8538 encapsulated NOTIFICATION Error subcode
           for the peer on this connection.";
        reference
          "RFC 8538: Notification Message Support for BGP Graceful
           Restart, Section 3.1.";
      }
    }

    /*
     * Containers
     */

    augment "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-protocol" {
      when "derived-from-or-self(rt:type, 'bgp')" {
        description
          "This augmentation is valid for a routing protocol
           instance of BGP.";
      }
      description
        "BGP protocol augmentation of ietf-routing module
         control-plane-protocol.";
      container bgp {
        description
          "Top-level configuration for the BGP router.";
        container global {
          presence "Enables global configuration of BGP";
          description
            "Global configuration for the BGP router.";
          leaf as {
            type inet:as-number;
            mandatory true;
```

```
            description
              "Local autonomous system number of the router.  Uses
               the 32-bit as-number type from the model in RFC 6991.";
          }
          leaf identifier {
            type yang:dotted-quad;
            description
              "BGP Identifier of the router - an unsigned 32-bit,
               non-zero integer that should be unique within an AS.
               The value of the BGP Identifier for a BGP speaker is
               determined upon startup and is the same for every local
               interface and BGP peer.";
            reference
              "RFC 6286: AS-Wide Unique BGP ID for BGP-4. Section 2.1";
          }
          container distance {
            description
              "Administrative distances (or preferences) assigned to
               routes received from different sources (external, and
               internal).";
            leaf external {
              type uint8 {
                range "1..255";
              }
              description
                "Administrative distances for routes learned from
                 external BGP (eBGP).";
            }
            leaf internal {
              type uint8 {
                range "1..255";
              }
              description
                "Administrative distances for routes learned from
                 internal BGP (iBGP).";
            }
          }
          container confederation {
            description
              "Configuration options specifying parameters when the
               local router is within an autonomous system which is
               part of a BGP confederation.";
            leaf enabled {
              type boolean;
              description
                "When this leaf is set to true it indicates that
                 the local-AS is part of a BGP confederation.";
            }
```

```
            leaf identifier {
              type inet:as-number;
              description
                "Confederation identifier for the autonomous system.";
            }
            leaf-list member-as {
              type inet:as-number;
              description
                "Remote autonomous systems that are to be treated
                 as part of the local confederation.";
            }
          }
          container graceful-restart {
            if-feature "bt:graceful-restart";
            description
              "Parameters relating the graceful restart mechanism for
               BGP.";
            reference
              "RFC 4724: Graceful Restart Mechanism for BGP.";
            uses graceful-restart-config;
          }
          uses global-group-use-multiple-paths;
          uses route-selection-options;
          container afi-safis {
            description
              "List of address-families associated with the BGP
               instance.";
            list afi-safi {
              key "name";
              description
                "AFI,SAFI configuration available for the
                 neighbor or group.";
              uses mp-afi-safi-config;
              uses state;
              container graceful-restart {
                if-feature "bt:graceful-restart";
                description
                  "Parameters relating to BGP graceful-restart";
                reference
                  "RFC 4724: Graceful Restart Mechanism for BGP.";
                uses mp-afi-safi-graceful-restart-config;
              }
              uses route-selection-options;
              uses structure-add-paths;
              uses global-group-use-multiple-paths;
              uses mp-all-afi-safi-list-contents;
            }
          }
```

```
          uses rt-pol:apply-policy-group;
          uses state;
        }

        container neighbors {
          description
            "Configuration for BGP neighbors.";

          list neighbor {
            key "remote-address";
            description
              "List of BGP neighbors configured on the local system,
               uniquely identified by remote IPv[46] address.";

            leaf remote-address {
              type inet:ip-address;
              description
                "The remote IP address of this entry’s BGP peer.";
            }

            leaf peer-group {
              type leafref {
                path "../../../peer-groups/peer-group/name";
              }
              description
                "The peer-group with which this neighbor is
                 associated.";
            }

            leaf local-address {
              type inet:ip-address;
              config false;
              description
                "The local IP address of this entry’s BGP connection.";
            }

            leaf local-port {
              type inet:port-number;
              config false;
              description
                "The local port for the TCP connection between
                 the BGP peers.";
            }

            leaf remote-port {
              type inet:port-number;
              config false;
              description
```

```
                "The remote port for the TCP connection
                 between the BGP peers.  Note that the
                 objects local-addr, local-port, remote-addr, and
                 remote-port provide the appropriate
                 reference to the standard MIB TCP
                 connection table.";
            }

            leaf peer-type {
              type bt:peer-type;
              config false;
              description
                "The type of peering session associated with this
                 neighbor.";
              reference
                "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
                           Section 1.1 for iBGP and eBGP.
                 RFC 5065: Autonomous System Configuration
                           for Confederation internal and external.";
            }

            leaf identifier {
              type yang:dotted-quad;
              config false;
              description
                "The BGP Identifier of this entry's BGP peer.
                 This entry MUST be 0.0.0.0 unless the
                 session state is in the openconfirm or the
                 established state.";
              reference
                "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
                           Section 4.2, 'BGP Identifier'.";
            }

            leaf dynamically-configured {
              type empty;
              config false;
              description
                "When present, this peer has been created
                dynamically.";
            }

            leaf enabled {
              type boolean;
              default "true";
              description
                "Whether the BGP peer is enabled. In cases where the
                 enabled leaf is set to false, the local system should
```

```
                not initiate connections to the neighbor, and should
                not respond to TCP connections attempts from the
                neighbor. If the state of the BGP session is
                Established at the time that this leaf is set to
                false, the BGP session should be ceased.

                A transition from 'false' to 'true' will cause
                the BGP Manual Start Event to be generated.
                A transition from 'true' to 'false' will cause
                the BGP Manual Stop Event to be generated.
                This parameter can be used to restart BGP peer
                connections. Care should be used in providing
                write access to this object without adequate
                authentication.";
              reference
                "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
                 Section 8.1.2.";
            }

            uses neighbor-group-config;

            container graceful-restart {
              if-feature "bt:graceful-restart";
              description
                "Parameters relating the graceful restart mechanism for
                 BGP";
              reference
                "RFC 4724: Graceful Restart Mechanism for BGP.";
              uses graceful-restart-config;
              leaf peer-restart-time {
                type uint16 {
                  range "0..4096";
                }
                config false;
                description
                  "The period of time (advertised by the peer) that the
                   peer expects a restart of a BGP session to take.";
              }

              leaf peer-restarting {
                type boolean;
                config false;
                description
                  "This flag indicates whether the remote neighbor is
                   currently in the process of restarting, and hence
                   received routes are currently stale.";
              }
```

```
              leaf local-restarting {
                type boolean;
                config false;
                description
                  "This flag indicates whether the local neighbor is
                   currently restarting. The flag is cleared after all
                   NLRI have been advertised to the peer, and the
                   End-of-RIB (EOR) marker has been cleared.";
              }

              leaf mode {
                type enumeration {
                  enum helper-only {
                    description
                      "The local router is operating in helper-only
                       mode, and hence will not retain forwarding state
                       during a local session restart, but will do so
                       during a restart of the remote peer";
                  }
                  enum bilateral {
                    description
                      "The local router is operating in both helper
                       mode, and hence retains forwarding state during
                       a remote restart, and also maintains forwarding
                       state during local session restart";
                  }
                  enum remote-helper {
                    description
                      "The local system is able to retain routes during
                       restart but the remote system is only able to
                       act as a helper";
                  }
                }
                config false;
                description
                  "This leaf indicates the mode of operation of BGP
                   graceful restart with the peer";
              }
            }

            container prefix-limit {
              description
                "Parameters relating to the prefix limit for the
                 AFI-SAFI";

              uses prefix-limit-config-common;

              uses prefix-limit-state-common;
```

```
            }

            container afi-safis {
              description
                "Per-address-family configuration parameters associated
                 with the neighbor";
              uses bgp-neighbor-afi-safi-list;
            }

            leaf session-state {
              type enumeration {
                enum idle {
                  description
                    "Neighbor is down, and in the Idle state of the
                     FSM.";
                }
                enum connect {
                  description
                    "Neighbor is down, and the session is waiting for
                     the underlying transport session to be
                     established.";
                }
                enum active {
                  description
                    "Neighbor is down, and the local system is awaiting
                     a connection from the remote peer.";
                }
                enum opensent {
                  description
                    "Neighbor is in the process of being established.
                     The local system has sent an OPEN message.";
                }
                enum openconfirm {
                  description
                    "Neighbor is in the process of being established.
                     The local system is awaiting a NOTIFICATION or
                     KEEPALIVE message.";
                }
                enum established {
                  description
                    "Neighbor is up - the BGP session with the peer is
                     established.";
                }
              }
              //  notification does not like a non-config statement.
              //  config false;
              description
                "The BGP peer connection state.";
```

```
              reference
                "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
                 Section 8.1.2.";
            }
            leaf last-established {
              type yang:date-and-time;
              config false;
              description
                "This timestamp indicates the time that the BGP session
                 last transitioned in or out of the Established state.
                 The value is the timestamp in seconds relative to the
                 Unix Epoch (Jan 1, 1970 00:00:00 UTC).

                 The BGP session uptime can be computed by clients as
                 the difference between this value and the current time
                 in UTC (assuming the session is in the Established
                 state, per the session-state leaf).";
              reference
                "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
                 Section 8.";
            }

            uses bgp-capabilities;

            container errors {
              config false;
              description
                "Information about BGP NOTIFICATION messages, sent and
                 received for this neighbor.";

              container received {
                description
                  "BGP NOTIFICATION message state received from this
                   neighbor.";
                uses bgp-errors-common;
                uses bgp-encapsulated-errors-common;
                uses bgp-errors-common-data;
              }
              container sent {
                description
                  "BGP NOTIFICATION message state sent to this
                   neighbor.";
                uses bgp-errors-common;
                uses bgp-encapsulated-errors-common;
                uses bgp-errors-common-data;
              }
            }
```

```
            container statistics {
              config false;
              description
                "Statistics per neighbor.";

              leaf established-transitions {
                type yang:zero-based-counter32;
                description
                  "Number of transitions to the Established state for
                   the neighbor session. This value is analogous to the
                   bgpPeerFsmEstablishedTransitions object from the
                   standard BGP-4 MIB";
                reference
                  "RFC 4273: Definitions of Managed Objects for
                   BGP-4, bgpPeerFsmEstablishedTransitions,
                   RFC 4271: A Border Gateway Protocol 4 (BGP-4),
                   Section 8.";
              }
              container messages {
                description
                  "Counters for BGP messages sent and received from the
                   neighbor";
                leaf total-received {
                  type yang:zero-based-counter32;
                  description
                    "Total number of BGP messages received from this
                     neighbor";
                  reference
                    "RFC 4273: Definitions of Managed Objects for
                     BGP-4, bgpPeerInTotalMessages.";
                }
                leaf total-sent {
                  type yang:zero-based-counter32;
                  description
                    "Total number of BGP messages sent do this
                     neighbor";
                  reference
                    "RFC 4273: Definitions of Managed Objects for
                     BGP-4, bgpPeerOutTotalMessages.";
                }
                leaf updates-received {
                  type yang:zero-based-counter32;
                  description
                    "Number of BGP UPDATE messages received from this
                     neighbor.";
                  reference
                    "RFC 4273: Definitions of Managed Objects for
                     BGP-4, bgpPeerInUpdates.";
```

```
                }
                leaf updates-sent {
                  type yang:zero-based-counter32;
                  description
                    "Number of BGP UPDATE messages sent to this
                     neighbor";
                  reference
                    "RFC 4273: Definitions of Managed Objects for
                     BGP-4, bgpPeerOutUpdates.";
                }
                leaf erroneous-updates-withdrawn {
                  type yang:zero-based-counter32;
                  config false;
                  description
                    "The number of BGP UPDATE messages for which the
                     treat-as-withdraw mechanism has been applied based
                     on erroneous message contents.";
                  reference
                    "RFC 7606: Revised Error Handling for BGP UPDATE
                     Messages, Section 2.";
                }
                leaf erroneous-updates-attribute-discarded {
                  type yang:zero-based-counter32;
                  config false;
                  description
                    "The number of BGP UPDATE messages for which the
                     attribute discard mechanism has been applied based
                     on erroneous message contents.";
                  reference
                    "RFC 7606: Revised Error Handling for BGP UPDATE
                     Messages, Section 2.";
                }
                leaf in-update-elapsed-time {
                  type yang:gauge32;
                  units "seconds";
                  description
                    "Elapsed time (in seconds) since the last BGP
                     UPDATE message was received from the peer.
                     Each time in-updates is incremented,
                     the value of this object is set to zero (0).";
                  reference
                    "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
                     Section 4.3
                     RFC 4271: A Border Gateway Protocol 4 (BGP-4),
                     Established state.";
                }
                leaf notifications-received {
                  type yang:zero-based-counter32;
```

```
                  description
                    "Number of BGP NOTIFICATION messages indicating an
                     error condition has occurred exchanged received
                     from this peer.";
                  reference
                    "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
                     Section 4.5.";
                }
                leaf notifications-sent {
                  type yang:zero-based-counter32;
                  description
                    "Number of BGP NOTIFICATION messages indicating an
                     error condition has occurred exchanged sent to
                     this peer.";
                  reference
                    "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
                     Section 4.5.";
                }
                leaf route-refreshes-received {
                  type yang:zero-based-counter32;
                  description
                    "Number of BGP ROUTE-REFRESH messages received from
                     this peer.";
                  reference
                    "RFC 2918: Route Refresh Capability for BGP-4.";
                }
                leaf route-refreshes-sent {
                  type yang:zero-based-counter32;
                  description
                    "Number of BGP ROUTE-REFRESH messages sent to
                     this peer.";
                  reference
                    "RFC 2918: Route Refresh Capability for BGP-4.";
                }
              }
              container queues {
                description
                  "Counters related to queued messages associated with
                   the BGP neighbor";
                leaf input {
                  type yang:gauge32;
                  description
                    "The number of messages received from the peer
                     currently queued";
                }
                leaf output {
                  type yang:gauge32;
                  description
```

```
                    "The number of messages queued to be sent to the
                     peer";
                }
              }
              action clear {
                if-feature "bt:clear-statistics";
                description
                  "Clear statistics action command.

                   Execution of this command should result in all the
                   counters to be cleared and set to 0.";

                input {
                  leaf clear-at {
                    type yang:date-and-time;
                    description
                      "Time when the clear action needs to be
                       executed.";
                  }
                }
                output {
                  leaf clear-finished-at {
                    type yang:date-and-time;
                    description
                      "Time when the clear action command completed.";
                  }
                }
              }
            }
          }

          notification established {
            leaf remote-address {
              type leafref {
                path "../../neighbor/remote-address";
              }
              description
                "IP address of the neighbor that went into established
                 state.";
            }
            description
              "The established event is generated
               when the BGP FSM enters the established state.";
          }

          notification backward-transition {
            leaf remote-addr {
              type leafref {
```

```
                path "../../neighbor/remote-address";
              }
              description
                "IP address of the neighbor that changed its state from
                 established state.";
            }
            container notification-received {
              description
                "If the backwards transition was caused by receiving a
                 BGP NOTIFICATION message, this is the information
                 received in the NOTIFICATION PDU.";

              uses bgp-errors-common;
              uses bgp-encapsulated-errors-common;
            }
            container notification-sent {
              description
                "If the backwards transition was caused by sending a
                 BGP NOTIFICATION message, this is the information
                 sent in the NOTIFICATION PDU.";

              uses bgp-errors-common;
              uses bgp-encapsulated-errors-common;
            }
            description
              "The backward-transition event is
               generated when the BGP FSM moves from a higher
               numbered state to a lower numbered state.";
          }
          action clear {
            if-feature "bt:clear-neighbors";
            description
              "Clear neighbors action.";

            input {
              choice operation {
                default operation-admin;
                description
                  "The type of operation for the clear action.";
                case operation-admin {
                  leaf admin {
                    type empty;
                    description
                      "Closes the Established BGP session with a BGP
                       NOTIFICATION message with the Administrative
                       Reset error subcode.";
                    reference
                      "RFC 4486 - Subcodes for BGP Cease Notification
```

```
                    Message.";
                }
              }
              case operation-hard {
                leaf hard {
                  type empty;
                  description
                    "Closes the Established BGP session with a BGP
                     NOTIFICATION message with the Hard Reset error
                     subcode.";
                  reference
                    "RFC 8538, Section 3 - Notification Message
                     Support for BGP Graceful Restart.";
                }
              }
              case operation-soft {
                leaf soft {
                  type empty;
                  description
                    "Re-sends the current Adj-Rib-Out to this
                     neighbor.";
                }
              }
              case operation-soft-inbound {
                leaf soft-inbound {
                  if-feature "bt:route-refresh";
                  type empty;
                  description
                    "Requests the Adj-Rib-In for this neighbor to be
                     re-sent using the BGP Route Refresh feature.";
                }
              }
            }

            leaf clear-at {
              type yang:date-and-time;
              description
                "Time when the clear action command needs to be
                 executed.";
            }
          }
          output {
            leaf clear-finished-at {
              type yang:date-and-time;
              description
                "Time when the clear action command completed.";
            }
          }
```

```
          }
        }

        container peer-groups {
          description
            "Configuration for BGP peer-groups";

          list peer-group {
            key "name";
            description
              "List of BGP peer-groups configured on the local system -
               uniquely identified by peer-group name";

            leaf name {
              type string;
              description
                "Name of the BGP peer-group";
            }

            uses neighbor-group-config;
            uses structure-dynamic-peers;

            container graceful-restart {
              if-feature "bt:graceful-restart";
              description
                "Parameters relating the graceful restart mechanism for
                 BGP";
              reference
                "RFC 4724: Graceful Restart Mechanism for BGP.";
              uses graceful-restart-config;
            }

            container prefix-limit {
              description
                "Parameters relating to the prefix limit for the
                 AFI-SAFI";

              uses prefix-limit-config-common;
            }

            container afi-safis {
              description
                "Per-address-family configuration parameters
                 associated with the peer-group.";
              list afi-safi {
                key "name";
                description
                  "AFI, SAFI configuration available for the
```

```
                   neighbor or group";
                 reference
                   "RFC 4724: Graceful Restart Mechanism for BGP.";
                 uses mp-afi-safi-config;
                 container graceful-restart {
                   if-feature "bt:graceful-restart";
                   description
                     "Parameters relating to BGP graceful-restart";
                   uses mp-afi-safi-graceful-restart-config;
                 }
                 uses structure-add-paths;
                 uses bgp-neighbor-use-multiple-paths;
                 uses mp-all-afi-safi-list-contents;
               }
             }
           }
         }

         uses rib;
       }
     }
   }
   <CODE ENDS>
```

7.1.2. BGP capabilities submodule

```
   <CODE BEGINS> file "ietf-bgp-capabilities@2023-07-05.yang"
   submodule ietf-bgp-capabilities {
     yang-version 1.1;
     belongs-to ietf-bgp {
       prefix bgp;
     }

     import iana-bgp-types {
       prefix bt;
       reference
         "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";

     }
     import iana-routing-types {
       prefix iana-rt-types;
       reference
         "RFC 8294: Common YANG Data Types for the Routing Area.";
     }
     import ietf-inet-types {
       prefix inet;
       reference
         "RFC 6991: Common YANG Data Types.";
```

```
     }

     organization
       "IETF IDR Working Group";
     contact
       "WG Web:   <http://datatracker.ietf.org/wg/idr>
        WG List:  <idr@ietf.org>

        Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                 Keyur Patel (keyur at arrcus.com),
                 Susan Hares (shares at ndzh.com,
                 Jeffrey Haas (jhaas at juniper.net).";

     description
       "This sub-module contains groupings that used for BGP
        Capabilities within the BGP module.

        Copyright (c) 2023 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 Revised 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 2023-07-05 {
       description
         "Initial Version";
       reference
         "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
     }

     grouping bgp-capabilities-common {
       description
         "BGP Capabilities that are used commonly by the
          advertised-capabilities and received-capabilities lists in the
```

```
          bgp-capabilities grouping.";

       leaf code {
         type uint8 {
           range 1..255;
         }
         description
           "BGP Capability Code";
         reference
           "RFC 5492: Capabilities Advertisement with BGP-4, Section
            4.";
       }
       leaf index {
         type uint8;
         description
           "Multiple BGP Capabilities with a given Capability Code may
            be advertised in a BGP OPEN message.  This index permits
            multiple such Capabilities to be represented in the YANG
            model.  It is RECOMMENDED that this index start at one (1)
            and increase by one for each additional instance of the
            Capability.";
         reference
           "RFC 5492: Capabilities Advertisement with BGP-4, Section
            4.";
       }
       leaf name {
         type identityref {
           base bt:bgp-capability;
         }
         description
           "When known, name carries the bgp-capability identity for the
            AFI/SAFI combination as used in the BGP YANG modules.";
       }

       container value {
         description
           "Some BGP Capabilities carry a Capability-specific Capability
            Value.";
         reference
           "RFC 5492: Capabilities Advertisement with BGP-4, Section
            4.";

         container mpbgp {
           when "../../name = 'bt:mp-bgp'";
           description
             "Multi-Protocol BGP-specific values.";
           reference
             "RFC 4760: Multiprotocol Extensions for BGP-4, Section 8.";
```

```
          leaf afi {
            type iana-rt-types:address-family;
            description
             "Address Family Identifier carried in the Multiprotocol
              Extensions Capability.";
            reference
              "RFC 4760: Multiprotocol Extensions for BGP-4, Section
               8.";
          }
          leaf safi {
            type iana-rt-types:bgp-safi;
            description
             "Subsequent Address Family Identifier carried in the
              Multiprotocol Extensions Capability.";
            reference
              "RFC 4760: Multiprotocol Extensions for BGP-4, Section
               8.";
          }
          leaf name {
            type identityref {
              base bt:afi-safi-type;
            }
            description
              "When known, name carries the BGP AFI-SAFI identity for
               the AFI/SAFI combination as used in the BGP YANG
               modules.";
          }
        }

        container graceful-restart {
          when "../../name = 'bt:graceful-restart'";
          description
            "BGP Graceful Restart-specific values.";
          reference
            "RFC 4724: Graceful Restart Mechanism for BGP.";

          leaf flags {
            type bt:graceful-restart-flags;
            description
              "Restart Flags advertised by the Graceful Restart
               Capability";
            reference
              "RFC 4724: Graceful Restart Mechanism for BGP, Section
               3.";
          }

          leaf unknown-flags {
            type bits {
```

```
              bit unknown-2 {
                position 2;
                description
                  "Bit 2 was received but is currently RESERVED.";
              }
              bit unknown-3 {
                position 3;
                description
                  "Bit 3 was received but is currently RESERVED.";
              }
            }
            description
              "When a bit is exchanged in the Graceful Restart Flags
               field that is unknown to this module, their bit position
               is rendered using the associated unknown bit.";
            reference
              "RFC 4724: Graceful Restart Mechanism for BGP, Section
               3.";
          }

          leaf restart-time {
            type bt:graceful-restart-time-type;
            description
              "The period of time (advertised by the peer) that the
               peer expects a restart of a BGP session to take.";
            reference
              "RFC 4724: Graceful Restart Mechanism for BGP, Section
               3.";
          }

          list afi-safis {
            description
              "List of AFI/SAFIs advertised by the BGP Graceful Restart
               Capability and their AFI/SAFI-specific Flags.";
            leaf afi {
              type iana-rt-types:address-family;
              description
                "Address Family advertised in the BGP Graceful Restart
                 Capability.";
              reference
                "RFC 4724: Graceful Restart Mechanism for BGP, Section
                 3.";
            }
            leaf safi {
              type iana-rt-types:bgp-safi;
              description
                "Subsequent Address Family advertised in the BGP
                 Graceful Restart Capability.";
```

```
              reference
                "RFC 4724: Graceful Restart Mechanism for BGP, Section
                 3.";
            }
            leaf afi-safi-flags {
              type bt:graceful-restart-flags-for-afi;
              description
                "Flags for Address Family advertised per-AFI/SAFI in
                 the BGP Graceful Restart Capability.";
              reference
                "RFC 4724: Graceful Restart Mechanism for BGP, Section
                 3.";
            }
            leaf afi-safi-unknown-flags {
              type bits {
                bit unknown-1 {
                  position 1;
                  description
                    "Bit 1 was received but is currently RESERVED.";
                }
                bit unknown-2 {
                  position 2;
                  description
                    "Bit 2 was received but is currently RESERVED.";
                }
                bit unknown-3 {
                  position 3;
                  description
                    "Bit 3 was received but is currently RESERVED.";
                }
                bit unknown-4 {
                  position 4;
                  description
                    "Bit 4 was received but is currently RESERVED.";
                }
                bit unknown-5 {
                  position 5;
                  description
                    "Bit 5 was received but is currently RESERVED.";
                }
                bit unknown-6 {
                  position 6;
                  description
                    "Bit 6 was received but is currently RESERVED.";
                }
                bit unknown-7 {
                  position 7;
                  description
```

```
                    "Bit 7 was received but is currently RESERVED.";
                }
              }
              description
                "When a bit is exchanged in the Graceful Restart Flags
                 for Address Family field that is unknown to this
                 module, their bit position is rendered using the
                 associated unknown bit.";
              reference
                "RFC 4724: Graceful Restart Mechanism for BGP, Section
                 3.";
            }
          }
        }

        container asn32 {
          when "../../name = 'bt:asn32'";
          description
            "Four-byte AS Number-specific values.";
          reference
            "RFC 6793: BGP Support for Four-Octet Autonomous System
             (AS) Number Space, Section 3.";

          leaf as {
            type inet:as-number;
            description
              "Four-byte AS Number carried by the Support for 4-octet
               AS number capability.";
            reference
              "RFC 6793: BGP Support for Four-Octet Autonomous System
               (AS) Number Space, Section 3.";
          }
        }

        container add-paths {
          when "../../name = 'bt:add-paths'";
          description
            "BGP add-paths-specific values.";
          reference
            "RFC 7911: Advertisement of Multiple Paths in BGP.";

          list afi-safis {
            description
              "List of AFI/SAFIs advertised by the BGP ADD-PATH
               Capability and their AFI/SAFI-specific mode.";
            leaf afi {
              type iana-rt-types:address-family;
              description
```

```
                "Address Family Identifier for an instance of the BGP
                 ADD-PATH Capability.";
              reference
                "RFC 7911: Advertisement of Multiple Paths in BGP,
                 Section 4.";
            }
            leaf safi {
              type iana-rt-types:bgp-safi;
              description
                "Subsequent Address Family Identifier for an instance
                 of the BGP ADD-PATH Capability.";
              reference
                "RFC 7911: Advertisement of Multiple Paths in BGP,
                 Section 4.";
            }
            leaf mode {
              type enumeration {
                enum receive {
                  value 1;
                  description
                    "The sender of the capability is able to receive
                     multiple paths from its peer.";
                  reference
                    "RFC 7911: Advertisement of Multiple Paths in BGP,
                     Section 4.";
                }
                enum send {
                  value 2;
                  description
                    "The sender of the capability is able to send
                     multiple paths to its peer.";
                  reference
                    "RFC 7911: Advertisement of Multiple Paths in BGP,
                     Section 4.";
                }
                enum receive-send {
                  value 3;
                  description
                    "The sender of the capability is able both receive
                     and send multiple paths for its peer.";
                  reference
                    "RFC 7911: Advertisement of Multiple Paths in BGP,
                     Section 4.";
                }
              }
              description
                "Send/Receive value for a per-AFI/SAFI instance of the
                 ADD-PATH Capability.";
```

```
               reference
                 "RFC 7911: Advertisement of Multiple Paths in BGP,
                  Section 4.";
             }
           }
         }
       }
     }

     grouping bgp-capabilities {
       description
         "Structural grouping used to include BGP Capabiliites for BGP
          neghbors.";

       container capabilities {
         config false;
         description
           "BGP Capabilities per-neighbor.";
         reference
           "RFC 5492: Capabilities Advertisement with BGP-4.";

         list advertised-capabilities {
           key "code index";
           description
             "List of advertised BGP capabilities, per-peer.
              Identified by the Capability Code and an index.  The
              index covers the case where the same BGP Capability
              may be advertised more than once.";

           uses bgp-capabilities-common;
         }

         list received-capabilities {
           key "code index";
           description
             "List of received BGP capabilities, per-peer.
              Identified by the Capability Code and an index.  The
              index covers the case where the same BGP Capability
              may be advertised more than once.";

           uses bgp-capabilities-common;
         }

         leaf-list negotiated-capabilities {
           type identityref {
             base bt:bgp-capability;
           }
           description
```

```
            "Negotiated BGP capabilities.";
        }
      }
    }
  }
  <CODE ENDS>
```

7.1.3. BGP common submodule

```
  <CODE BEGINS> file "ietf-bgp-common@2023-07-05.yang"
  submodule ietf-bgp-common {
    yang-version 1.1;
    belongs-to ietf-bgp {
      prefix bgp;
    }

    import iana-bgp-types {
      prefix bt;
      reference
        "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
    }
    import iana-bgp-community-types {
      prefix bct;
      reference
        "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
    }
    import ietf-interfaces {
      prefix if;
      reference
        "RFC 7223: A YANG Data Model for Interface Management.";
    }
    import ietf-key-chain {
      prefix key-chain;
      reference
        "RFC 8177: YANG Data Model for Key Chains.";
    }
    import ietf-inet-types {
      prefix inet;
      reference
        "RFC 6991: Common YANG Data Types.";
    }
    import ietf-routing-policy {
      prefix rt-pol;
      reference
        "RFC 9067: A YANG Data Model for Routing Policy.";
    }
    import ietf-yang-types {
      prefix yang;
```

```
      reference
        "RFC 6991: Common YANG Data Types.";
    }
    import ietf-bfd-types {
      prefix bfd-types;
      reference
        "RFC 9127: YANG Data Model for Bidirectional Forward
         Detection.";
    }
    import ietf-tcp {
      prefix tcp;
      reference
        "I-D.ietf-tcpm-yang-tcp: Transmission Control Protocol (TCP)
         YANG Model.";
    }
    include ietf-bgp-common-structure {
      revision-date 2023-07-05;
    }

    organization
      "IETF IDR Working Group";
    contact
      "WG Web:   <http://datatracker.ietf.org/wg/idr>
       WG List:  <idr@ietf.org>

       Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                Keyur Patel (keyur at arrcus.com),
                Susan Hares (shares at ndzh.com,
                Jeffrey Haas (jhaas at juniper.net).";

    description
      "This sub-module contains common groupings that are common across
       multiple contexts within the BGP module. That is to say that
       they may be application to a subset of global, peer-group, or
       neighbor contexts.

       Copyright (c) 2023 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 Revised 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 2023-07-05 {
       description
         "Initial Version";
       reference
         "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
     }

     grouping neighbor-group-timers {
       description
         "Config parameters related to timers associated with the BGP
          peer";
       leaf connect-retry-interval {
         type uint16 {
           range "1..max";
         }
         units "seconds";
         default "120";
         description
           "Time interval (in seconds) for the ConnectRetryTimer. The
            suggested value for this timer is 120 seconds.";
         reference
           "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
            Section 8.2.2.";
       }
       leaf hold-time {
         type uint16 {
           range "0 | 3..65535";
         }
         units "seconds";
         default "90";
         description
           "Time interval (in seconds) for the HoldTimer established
            with the peer.  When read as operational data (ro), the
            value of this object is calculated by this BGP speaker,
            using the smaller of the values in hold-time that was
            configured (rw) in the running datastore and the Hold Time
            received in the OPEN message.

            This value must be at least three seconds
            if it is not zero (0).
```

```
            If the Hold Timer has not been established
            with the peer this object MUST have a value
            of zero (0).

            If the configured value of hold-time object was
            a value of (0), then when read this object MUST have a
            value of (0) also.";
          reference
            "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.2,
             RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 10.";
        }
        leaf negotiated-hold-time {
          type uint16;
          config false;
          description
            "The negotiated hold-time for the BGP session";
          reference
            "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.2,
             RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 10.";
        }
        leaf keepalive {
          type uint16 {
            range "0..21845";
          }
          units "seconds";
          description
            "When used as a configuration (rw) value, this Time interval
             (in seconds) for the KeepAlive timer configured for this BGP
             speaker with this peer.  A reasonable maximum value for this
             timer would be one-third of the configured hold-time.

             In the absence of explicit configuration of the keepalive
             value, operationally it SHOULD have a value of one-third of
             the negotiated hold-time.

             If the value of this object is zero (0), no periodic
             KEEPALIVE messages are sent to the peer after the BGP
             connection has been established.

             The actual time interval for the KEEPALIVE messages is
             indicated by operational value of keepalive.";
          reference
            "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.4,
             RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 10.";
        }
        leaf min-as-origination-interval {
          type uint16 {
            range "0..max";
```

```
        }
        units "seconds";
        description
          "Time interval (in seconds) for the MinASOriginationInterval
           timer. The suggested value for this timer is 15 seconds.";
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
           9.2.1.2,
           RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 10.";
      }
      leaf min-route-advertisement-interval {
        type uint16 {
          range "0..max";
        }
        units "seconds";
        description
          "Time interval (in seconds) for the
           MinRouteAdvertisementInterval timer.
           The suggested value for this timer is 30
           seconds for EBGP connections and 5
           seconds for IBGP connections.";
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
           9.2.1.1,
           RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 10.";
      }
    }

    grouping bgp-neighbor-use-multiple-paths {
      description
        "Multi-path configuration and state applicable to a BGP
         neighbor";
      container use-multiple-paths {
        description
          "Parameters related to the use of multiple-paths for the same
           NLRI when they are received only from this neighbor";
        leaf enabled {
          type boolean;
          default "false";
          description
            "Whether the use of multiple paths for the same NLRI is
             enabled for the neighbor.";
        }
        container ebgp {
          description
            "Multi-path configuration for eBGP";
          leaf allow-multiple-as {
            type boolean;
```

```
            default "false";
            description
              "Allow multi-path to use paths from different neighboring
               ASes. The default is to only consider multiple paths
               from the same neighboring AS.";
          }
        }
      }
    }

    grouping neighbor-group-config {
      description
        "Neighbor level configuration items.";
      leaf peer-as {
        type inet:as-number;
        description
          "AS number of the peer.";
      }
      leaf local-as {
        type inet:as-number;
        description
          "The local autonomous system number that is to be used when
           establishing sessions with the remote peer or peer group, if
           this differs from the global BGP router autonomous system
           number.";
      }
      leaf remove-private-as {
        type bt:remove-private-as-option;
        description
          "When this leaf is specified, remove private AS numbers from
           updates sent to peers.";
      }
      container route-flap-damping {
        if-feature "bt:damping";
        leaf enable {
          type boolean;
          default "false";
          description
            "Enable route flap damping.";
        }
        leaf suppress-above {
          type decimal64 {
            fraction-digits 1;
          }
          default "3.0";
          description
            "This is the value of the instability metric at which
             route suppression takes place. A route is not installed
```

```
             in the forwarding information base (FIB), or announced
             even if it is reachable during the period that it is
             suppressed.";
         }
         leaf reuse-above {
           type decimal64 {
             fraction-digits 1;
           }
           default "2.0";
           description
             "This is the value of the instability metric at which a
              suppressed route becomes unsuppressed if it is reachable
              but currently suppressed. The value assigned to
              reuse-below must be less than suppress-above.";
         }
         leaf max-flap {
           type decimal64 {
             fraction-digits 1;
           }
           default "16.0";
           description
             "This is the upper limit of the instability metric. This
              value must be greater than the larger of 1 and
              suppress-above.";
         }
         leaf reach-decay {
           type uint32;
           units "seconds";
           default "300";
           description
             "This value specifies the time desired for the instability
              metric value to reach one-half of its current value when
              the route is reachable. This half-life value determines
              the rate at which the metric value is decayed. A smaller
              half-life value makes a suppressed route reusable sooner
              than a larger value.";
         }
         leaf unreach-decay {
           type uint32;
           units "seconds";
           default "900";
           description
             "This value acts the same as reach-decay except that it
              specifies the rate at which the instability metric is
              decayed when a route is unreachable. It should have a
              value greater than or equal to reach-decay.";
         }
         leaf keep-history {
```

```
          type uint32;
          units "seconds";
          default "1800";
          description
            "This value specifies the period over which the route
             flapping history is to be maintained for a given route.
             The size of the configuration arrays described below is
             directly affected by this value.";
        }
        description
          "Routes learned via BGP are subject to weighted route
           dampening.";
      }
      leaf-list send-community {
        if-feature "bct:send-communities";
        type identityref {
          base "bct:send-community-feature";
        }
        description
          "When supported, this tells the router to propagate any
           prefixes that are attached to these community-types.";
      }
      leaf description {
        type string;
        description
          "An optional textual description (intended primarily for use
           with a peer or group";
      }

      container timers {
        description
          "Timers related to a BGP neighbor";
        uses neighbor-group-timers;
      }

      container transport {
        description
          "Transport session parameters for the BGP neighbor";
        uses neighbor-group-transport-config;
      }

      leaf treat-as-withdraw {
        type boolean;
        default "false";
        description
          "Specify whether erroneous UPDATE messages for which the NLRI
           can be extracted are treated as though the NLRI is withdrawn
           - avoiding session reset";
```

```
        reference
          "RFC 7606: Revised Error Handling for BGP UPDATE Messages.";
      }

      container logging-options {
        description
          "Logging options for events related to the BGP neighbor or
           group";
        leaf log-neighbor-state-changes {
          type boolean;
          default "true";
          description
            "Configure logging of peer state changes.  Default is to
             enable logging of peer state changes.

             Note: Documenting demotion from ESTABLISHED state is
                   desirable, but documenting all backward transitions
                   is problematic, and should be avoided.";
        }
      }

      uses structure-neighbor-group-route-reflector;
      uses structure-neighbor-group-as-path-options;
      uses structure-add-paths;
      uses bgp-neighbor-use-multiple-paths;
      uses rt-pol:apply-policy-group;
    }

    grouping neighbor-group-transport-config {
      description
        "Configuration parameters relating to the transport protocol
         used by the BGP session to the peer.";

      leaf local-address {
        type union {
          type inet:ip-address;
          type if:interface-ref;
        }
        description
          "Set the local IP (either IPv4 or IPv6) address to use for
           the session when sending BGP update messages. This may be
           expressed as either an IP address or reference to the name
           of an interface.";
      }

      leaf tcp-mss {
        type tcp:mss;
        description
```

```
          "Maximum Segment Size (MSS) desired on this connection.
           Note, the 'effective send MSS' can be smaller than
           what is configured here.";
         reference
           "RFC 9293: Transmission Control Protocol (TCP)
                      Specification.";
       }

       leaf mtu-discovery {
         type boolean;
         default "true";
         description
           "Turns path mtu discovery for BGP TCP sessions on (true) or
            off (false).";
         reference
           "RFC 1191: Path MTU discovery.";
       }

       container ebgp-multihop {
         description
           "eBGP multi-hop parameters for the BGP peer-group";
         leaf enabled {
           type boolean;
           default "false";
           description
             "When enabled, the referenced group or neighbors are
              permitted to be indirectly connected - including cases
              where the TTL can be decremented between the BGP peers";
         }
         leaf multihop-ttl {
           type uint8;
           description
             "Time-to-live value to use when packets are sent to the
              referenced group or neighbors and ebgp-multihop is
              enabled";
         }
       }

       leaf passive-mode {
         type boolean;
         default "false";
         description
           "Wait for peers to issue requests to open a BGP session,
            rather than initiating sessions from the local router.";
       }

       leaf ttl-security {
         if-feature "bt:ttl-security";
```

```
        type uint8;
        default "255";
        description
          "BGP Time To Live (TTL) security check.";
        reference
          "RFC 5082: The Generalized TTL Security Mechanism (GTSM),
           RFC 7454: BGP Operations and Security.";
      }

      container secure-session {
        must "enabled = 'false' or " +
             "(enabled = 'true' and options)" {
          error-message "When secure-session is enabled " +
            "secure-session options MUST be set";
        }
        description
          "Container for describing how a particular BGP session
           is to be secured.";

        leaf enabled {
          type boolean;
          default "false";
          description
            "When set to true, session is secured with the
             configured options.";
        }

        container options {
          description
            "Options for securing the BGP session.";
          choice option {
            case ao {
              leaf ao-keychain {
                type key-chain:key-chain-ref;
                description
                  "Reference to the key chain that will be used by this
                   model. Applicable for TCP-AO and TCP-MD5 only";
                reference
                  "RFC 8177: YANG Data Model for Key Chains.";
              }
              description
                "Uses TCP-AO to secure the session. Parameters for
                 those are defined as a grouping in the TCP YANG
                 model.";
              reference
                "RFC 5925: The TCP Authentication Option.";
            }
```

```
            case md5 {
              leaf md5-keychain {
                type key-chain:key-chain-ref;
                description
                  "Reference to the key chain that will be used by this
                   model. Applicable for TCP-AO and TCP-MD5 only";
                reference
                  "RFC 8177: YANG Data Model for Key Chains.";
              }
              description
                "Uses TCP-MD5 to secure the session. Parameters for
                 those are defined as a grouping in the TCP YANG
                 model.";
              reference
                "RFC 5925: The TCP Authentication Option.";
            }

            case ipsec {
              leaf sa {
                type string;
                description
                  "Security Association (SA) name.";
              }
              description
                "Currently, the IPsec/IKE YANG model has no
                 grouping defined that this model can use. When
                 such a grouping is defined, this model can import
                 the grouping to add the key parameters
                 needed to kick of IKE.";
            }

            description
              "Choice of authentication options.";
          }
        }
      }

      container bfd {
        if-feature "bt:bfd";
        uses bfd-types:client-cfg-parms;
        description
          "BFD client configuration.";
        reference
          "RFC 9127: YANG Data Model for Bidirectional Forwarding
           Detection.";
      }
    }
```

```
     grouping graceful-restart-config {
       description
         "Configuration parameters relating to BGP graceful restart.";
       leaf enabled {
         type boolean;
         default "false";
         description
           "Enable or disable the graceful-restart capability.";
       }
       leaf restart-time {
         type bt:graceful-restart-time-type;
         description
           "Estimated time (in seconds) for the local BGP speaker to
            restart a session. This value is advertise in the graceful
            restart BGP capability.  This is a 12-bit value, referred to
            as Restart Time in RFC4724.  Per RFC4724, the suggested
            default value is <= the hold-time value.";
         reference
           "RFC 4724: Graceful Restart Mechanism for BGP.";
       }
       leaf stale-routes-time {
         type uint32;
         description
           "An upper-bound on the time that stale routes will be
            retained by a router after a session is restarted. If an
            End-of-RIB (EOR) marker is received prior to this timer
            expiring, stale-routes will be flushed upon its receipt - if
            no EOR is received, then when this timer expires stale paths
            will be purged. This timer is referred to as the
            Selection_Deferral_Timer in RFC4724";
         reference
           "RFC 4724: Graceful Restart Mechanism for BGP.";
       }
       leaf helper-only {
         type boolean;
         default "true";
         description
           "Enable graceful-restart in helper mode only. When this leaf
            is set, the local system does not retain forwarding its own
            state during a restart, but supports procedures for the
            receiving speaker, as defined in RFC4724.";
         reference
           "RFC 4724: Graceful Restart Mechanism for BGP.";
       }
     }

     grouping global-group-use-multiple-paths {
       description
```

```
        "Common grouping used for both global and groups which provides
         configuration and state parameters relating to use of multiple
         paths";
      container use-multiple-paths {
        description
          "Parameters related to the use of multiple paths for the
           same NLRI";
        leaf enabled {
          type boolean;
          default "false";
          description
            "Whether the use of multiple paths for the same NLRI is
             enabled for the neighbor. This value is overridden by any
             more specific configuration value.";
        }
        container ebgp {
          description
            "Multi-Path parameters for eBGP";
          leaf allow-multiple-as {
            type boolean;
            default "false";
            description
              "Allow multi-path to use paths from different neighboring
               ASes.  The default is to only consider multiple paths
               from the same neighboring AS.";
          }
          leaf maximum-paths {
            type uint32;
            default "1";
            description
              "Maximum number of parallel paths to consider when using
               BGP multi-path. The default is use a single path.";
          }
        }
        container ibgp {
          description
            "Multi-Path parameters for iBGP";
          leaf maximum-paths {
            type uint32;
            default "1";
            description
              "Maximum number of parallel paths to consider when using
               iBGP multi-path. The default is to use a single path";
          }
        }
      }
    }
```

```
     grouping route-selection-options {
       description
         "Configuration and state relating to route selection options";
       container route-selection-options {
         description
           "Parameters relating to options for route selection";
         leaf always-compare-med {
           type boolean;
           default "false";
           description
             "Compare multi-exit discriminator (MED) value from
              different ASes when selecting the best route.  The default
              behavior is to only compare MEDs for paths received from
              the same AS.";
         }
         leaf ignore-as-path-length {
           type boolean;
           default "false";
           description
             "Ignore the AS path length when selecting the best path.
              The default is to use the AS path length and prefer paths
              with a shorter length.";
         }
         leaf external-compare-router-id {
           type boolean;
           default "true";
           description
             "When comparing similar routes received from external BGP
              peers, use the router-id as a criterion to select the
              active path.";
         }
         leaf advertise-inactive-routes {
           type boolean;
           default "false";
           description
             "Advertise inactive routes to external peers.  The default
              is to only advertise active routes.";
           reference
             "I-D.ietf-idr-best-external: Advertisement of the best
              external route in BGP.";
         }
         leaf enable-aigp {
           type boolean;
           default "false";
           description
             "Flag to enable sending / receiving accumulated IGP
              attribute in routing updates";
           reference
```

```
            "RFC 7311: AIGP Metric Attribute for BGP.";
        }
        leaf ignore-next-hop-igp-metric {
          type boolean;
          default "false";
          description
            "Ignore the IGP metric to the next-hop when calculating BGP
             best-path. The default is to select the route for which
             the metric to the next-hop is lowest";
        }
        leaf enable-med {
          type boolean;
          default "false";
          description
            "Flag to enable sending/receiving of MED metric attribute
             in routing updates.";
        }
        container med-plus-igp {
          leaf enabled {
            type boolean;
            default "false";
            description
              "When enabled allows BGP to use MED and IGP values
               defined below to determine the optimal route.";
            reference
              "RFC 4451: BGP MED Considerations.";
          }
          leaf igp-multiplier {
            type uint16;
            default 1;
            description
              "Specifies an IGP cost multiplier.";
            reference
              "RFC 4451: BGP MED Considerations.";
          }
          leaf med-multiplier {
            type uint16;
            default 1;
            description
              "Specifies a MED multiplier.";
            reference
              "RFC 4451: BGP MED Considerations.";
          }
          description
            "The med-plus-igp option enables BGP to use the sum of
             MED multiplied by a MED multiplier and IGP cost multiplied
             by IGP cost multiplier to select routes when MED is
             required to determine the optimal route.";
```

```
        }
      }
    }

    grouping state {
      description
        "Grouping containing common counters relating to prefixes and
         paths";

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

        leaf total-paths {
          type yang:gauge32;
          description
            "Total number of BGP paths (BGP routes) within the
             context";
        }

        leaf total-prefixes {
          type yang:gauge32;
          description
            "Total number of BGP prefixes (destinations) received
             within the context";
        }
      }
    }
  }
  <CODE ENDS>
```

7.1.4. BGP common multiprotocol submodule

```
  <CODE BEGINS> file "ietf-bgp-common-multiprotocol@2023-07-05.yang"
  submodule ietf-bgp-common-multiprotocol {
    yang-version 1.1;
    belongs-to ietf-bgp {
      prefix bgp;
    }

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

```
     }
     import ietf-routing-policy {
       prefix rt-pol;
     }

     // meta

     organization
       "IETF IDR Working Group";
     contact
       "WG Web:   <http://datatracker.ietf.org/wg/idr>
        WG List:  <idr@ietf.org>

        Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                 Keyur Patel (keyur at arrcus.com),
                 Susan Hares (shares at ndzh.com),
                 Jeffrey Haas (jhaas at juniper.net).";

     description
       "This sub-module contains groupings that are related to support
        for multiple protocols in BGP. The groupings are common across
        multiple contexts.

        Copyright (c) 2023 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 Revised 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 2023-07-05 {
       description
         "Initial Version";
       reference
         "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
```

```
     }

     grouping prefix-limit-config-common {
       description
         "Common configuration for prefix-limit feature.";

       leaf max-prefixes {
         type uint32;
         description
           "Maximum number of prefixes that will be accepted from the
            neighbor";
       }
       leaf warning-threshold-pct {
         type rt-types:percentage;
         description
           "Threshold on number of prefixes that can be received from
            a neighbor before generation of warning messages or log
            entries. Expressed as a percentage of max-prefixes";
       }
       leaf teardown {
         type boolean;
         default false;
         description
           "When 'true', tear down the BGP session when the maximum
            prefix limit is exceeded.  When 'false', only log a
            warning when the maximum prefix limit is exceeded.";
       }
       leaf idle-time {
         type union {
           type uint32;
           type enumeration {
             enum forever {
               description
                 "Idle the peer until manually reset.";
             }
           }
         }
         units "seconds";
         description
           "Time interval in seconds after which the BGP session is
            re-established after being torn down due to exceeding the
            max-prefix limit.";
       }
     }

     grouping prefix-limit-state-common {
       description
         "Common operational state for prefix-limit feature.";
```

```
      leaf prefix-limit-exceeded {
        type boolean;
        config false;
        description
          "'true' when the prefix-limit has been exceeded for this
           scope.";
      }
    }


    grouping mp-afi-safi-graceful-restart-config {
      description
        "BGP graceful restart parameters that apply on a per-AFI-SAFI
         basis";
      leaf enabled {
        type boolean;
        must ". = ../../../../graceful-restart/enabled";
        default "false";
        description
          "This leaf indicates whether graceful-restart is enabled for
           this AFI-SAFI.";
        reference
          "RFC 4724: Graceful Restart Mechanism for BGP.";
      }
    }

    grouping mp-afi-safi-config {
      description
        "Configuration parameters used for all BGP AFI-SAFIs";
      leaf name {
        type identityref {
          base bt:afi-safi-type;
        }
        description
          "AFI,SAFI";
      }
      leaf enabled {
        type boolean;
        default "false";
        description
          "This leaf indicates whether this AFI,SAFI is enabled for
           the neighbor or group";
      }
    }

    grouping mp-all-afi-safi-list-contents {
      description
        "A common grouping used for contents of the list that is used
```

```
         for AFI-SAFI entries";
       // import and export policy included for the afi/safi
       uses rt-pol:apply-policy-group;
       container ipv4-unicast {
         when "../name = 'bt:ipv4-unicast'" {
           description
             "Include this container for IPv4 Unicast specific
              configuration";
         }
         description
           "IPv4 unicast configuration options";
         // include common IPv[46] unicast options
         uses mp-ipv4-ipv6-unicast-common;
         // placeholder for IPv4 unicast specific configuration
       }
       container ipv6-unicast {
         when "../name = 'bt:ipv6-unicast'" {
           description
             "Include this container for IPv6 Unicast specific
              configuration";
         }
         description
           "IPv6 unicast configuration options";
         // include common IPv[46] unicast options
         uses mp-ipv4-ipv6-unicast-common;
         // placeholder for IPv6 unicast specific configuration
         // options
       }
       container ipv4-labeled-unicast {
         when "../name = 'bt:ipv4-labeled-unicast'" {
           description
             "Include this container for IPv4 Labeled Unicast specific
              configuration";
         }
         description
           "IPv4 Labeled Unicast configuration options";
         uses mp-all-afi-safi-common;
         // placeholder for IPv4 Labeled Unicast specific config
         // options
       }
       container ipv6-labeled-unicast {
         when "../name = 'bt:ipv6-labeled-unicast'" {
           description
             "Include this container for IPv6 Labeled Unicast specific
              configuration";
         }
         description
           "IPv6 Labeled Unicast configuration options";
```

```
        uses mp-all-afi-safi-common;
        // placeholder for IPv6 Labeled Unicast specific config
        // options.
      }
      container l3vpn-ipv4-unicast {
        when "../name = 'bt:l3vpn-ipv4-unicast'" {
          description
            "Include this container for IPv4 Unicast L3VPN specific
             configuration";
        }
        description
          "Unicast IPv4 L3VPN configuration options";
        // include common L3VPN configuration options
        uses mp-l3vpn-ipv4-ipv6-unicast-common;
        // placeholder for IPv4 Unicast L3VPN specific config options.
      }
      container l3vpn-ipv6-unicast {
        when "../name = 'bt:l3vpn-ipv6-unicast'" {
          description
            "Include this container for unicast IPv6 L3VPN specific
             configuration";
        }
        description
          "Unicast IPv6 L3VPN configuration options";
        // include common L3VPN configuration options
        uses mp-l3vpn-ipv4-ipv6-unicast-common;
        // placeholder for IPv6 Unicast L3VPN specific configuration
        // options
      }
      container l3vpn-ipv4-multicast {
        when "../name = 'bt:l3vpn-ipv4-multicast'" {
          description
            "Include this container for multicast IPv6 L3VPN specific
             configuration";
        }
        description
          "Multicast IPv4 L3VPN configuration options";
        // include common L3VPN multicast options
        uses mp-l3vpn-ipv4-ipv6-multicast-common;
        // placeholder for IPv4 Multicast L3VPN specific configuration
        // options
      }
      container l3vpn-ipv6-multicast {
        when "../name = 'bt:l3vpn-ipv6-multicast'" {
          description
            "Include this container for multicast IPv6 L3VPN specific
             configuration";
        }
```

```
        description
          "Multicast IPv6 L3VPN configuration options";
        // include common L3VPN multicast options
        uses mp-l3vpn-ipv4-ipv6-multicast-common;
        // placeholder for IPv6 Multicast L3VPN specific configuration
        // options
      }
      container l2vpn-vpls {
        when "../name = 'bt:l2vpn-vpls'" {
          description
            "Include this container for BGP-signalled VPLS specific
             configuration";
        }
        description
          "BGP-signalled VPLS configuration options";
        // include common L2VPN options
        uses mp-l2vpn-common;
        // placeholder for BGP-signalled VPLS specific configuration
        // options
      }
      container l2vpn-evpn {
        when "../name = 'bt:l2vpn-evpn'" {
          description
            "Include this container for BGP EVPN specific
             configuration";
        }
        description
          "BGP EVPN configuration options";
        // include common L2VPN options
        uses mp-l2vpn-common;
        // placeholder for BGP EVPN specific configuration options
      }
    }

    // Common groupings across multiple AFI,SAFIs

    grouping mp-all-afi-safi-common {
      description
        "Grouping for configuration common to all AFI,SAFI";
      container prefix-limit {
        description
          "Parameters relating to the prefix limit for the AFI-SAFI";

        uses prefix-limit-config-common;

        uses prefix-limit-state-common;
      }
    }
```

```
     grouping mp-ipv4-ipv6-unicast-common {
       description
         "Common configuration that is applicable for IPv4 and IPv6
          unicast";
       // include common afi-safi options.
       uses mp-all-afi-safi-common;
       // configuration options that are specific to IPv[46] unicast
       leaf send-default-route {
         type boolean;
         default "false";
         description
           "If set to true, send the default-route to the neighbor(s)";
       }
     }

     grouping mp-l3vpn-ipv4-ipv6-unicast-common {
       description
         "Common configuration applied across L3VPN for IPv4
          and IPv6";
       // placeholder -- specific configuration options that are generic
       // across IPv[46] unicast address families.
       uses mp-all-afi-safi-common;
     }

     grouping mp-l3vpn-ipv4-ipv6-multicast-common {
       description
         "Common configuration applied across L3VPN for IPv4
          and IPv6";
       // placeholder -- specific configuration options that are
       // generic across IPv[46] multicast address families.
       uses mp-all-afi-safi-common;
     }

     grouping mp-l2vpn-common {
       description
         "Common configuration applied across L2VPN address
          families";
       // placeholder -- specific configuration options that are
       // generic across L2VPN address families
       uses mp-all-afi-safi-common;
     }

     // Config groupings for common groups

     grouping mp-all-afi-safi-common-prefix-limit-config {
       description
         "Configuration parameters relating to prefix-limits for an
          AFI-SAFI";
```

```
    }
  }
  <CODE ENDS>
```

7.1.5. BGP common structure submodule

```
  <CODE BEGINS> file "ietf-bgp-common-structure@2023-07-05.yang"
  submodule ietf-bgp-common-structure {
    yang-version 1.1;
    belongs-to ietf-bgp {
      prefix bgp;
    }

    import ietf-routing-policy {
      prefix rt-pol;
      reference
        "RFC 9067: A YANG Data Model for Routing Policy Management.";
    }
    import iana-bgp-types {
      prefix bt;
      reference
        "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
    }
    import ietf-inet-types {
      prefix inet;
      reference
        "RFC 6991: Common YANG Data Types.";
    }
    include ietf-bgp-common-multiprotocol;

    // meta

    organization
      "IETF IDR Working Group";
    contact
      "WG Web:   <http://datatracker.ietf.org/wg/idr>
       WG List:  <idr@ietf.org>

       Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                Keyur Patel (keyur at arrcus.com),
                Susan Hares (shares at ndzh.com),
                Jeffrey Haas (jhaas at juniper.net).";

    description
      "This sub-module contains groupings that are common across
       multiple BGP contexts and provide structure around other
       primitive groupings.
```

```
        Copyright (c) 2023 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 Revised 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 2023-07-05 {
       description
         "Initial Version";
       reference
         "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
     }

     grouping structure-neighbor-group-route-reflector {
       description
         "Structural grouping used to include route reflector
          configuration and state for both BGP neighbors and peer
          groups";
       container route-reflector {
         description
           "Route reflector parameters for the BGP peer-group";
         reference
           "RFC 4456: BGP Route Reflection.";
         leaf cluster-id {
           type bt:rr-cluster-id-type;
           description
             "Route Reflector cluster id to use when local router is
              configured as a route reflector.  Commonly set at the
              group level, but allows a different cluster id to be set
              for each neighbor.";
           reference
             "RFC 4456: BGP Route Reflection: An Alternative to
                        Full Mesh.";
         }
```

```
        leaf client {
          type boolean;
          default "false";
          description
            "Configure the neighbor as a route reflector client.";
          reference
            "RFC 4456: BGP Route Reflection: An Alternative to
                       Full Mesh.";
        }
      }
    }

    grouping structure-neighbor-group-as-path-options {
      description
        "Structural grouping used to include AS_PATH manipulation
         configuration and state for both BGP neighbors and peer
         groups";
      container as-path-options {
        description
          "AS_PATH manipulation parameters for the BGP neighbor or
           group";
        leaf allow-own-as {
          type uint8;
          default "0";
          description
            "Specify the number of occurrences of the local BGP
             speaker's AS that can occur within the AS_PATH before it
             is rejected as looped.";
        }
        leaf replace-peer-as {
          type boolean;
          default "false";
          description
            "Replace occurrences of the peer's AS in the AS_PATH with
             the local autonomous system number";
        }
        leaf disable-peer-as-filter {
          type boolean;
          default "false";
          description
            "When set to true, the system advertises routes to a peer
             even if the peer's AS was in the AS path.  The default
             behavior (false) suppresses advertisements to peers if
             their AS number is in the AS path of the route.";
        }
      }
    }
```

```
     grouping structure-add-paths {
       description
         "Structural grouping used to include ADD-PATHs configuration
          and state.";
       container add-paths {
         if-feature "bt:add-paths";
         description
           "Parameters relating to the advertisement and receipt of
            multiple paths for a single NLRI (add-paths)";
         reference
           "RFC 7911: Advertisements of Multiple Paths in BGP.";
         leaf receive {
           type boolean;
           default "false";
           description
             "Enable ability to receive multiple path advertisements for
              an NLRI from the neighbor or group";
         }
         choice send {
           description
             "Choice of sending the max. number of paths or to send
              all.";
           case max {
             leaf max {
               type uint8;
               description
                 "The maximum number of paths to advertise to neighbors
                  for a single NLRI";
             }
           }
           case all {
             leaf all {
               type empty;
               description
                 "Send all the path advertisements to neighbors for a
                  single NLRI.";
             }
           }
         }
         leaf eligible-prefix-policy {
           type leafref {
             path "/rt-pol:routing-policy/rt-pol:policy-definitions/"
                + "rt-pol:policy-definition/rt-pol:name";
           }
           description
             "A reference to a routing policy which can be used to
              restrict the prefixes for which add-paths is enabled";
         }
```

```
      }
    }

    grouping structure-dynamic-peers {
      description
        "Structural grouping to contain dyamic BGP peers. Dynamic
         peers are configured from a list of IP prefixes matching the
         IP source address of the dynamic peer.";

      container dynamic-peers {
        description
          "Configuration and operational state for dynamically
           configured peers.";

        list dynamic-peer-list {
          key "prefix";
          description
            "List of peers by IP prefix for this configuration scope
             that are dynamically configured.";

          leaf prefix {
            type inet:ip-prefix;
            description
              "Prefix for dynamic peer at this configuration scope.";
          }
        }
      }
    }
  }
  <CODE ENDS>
```

7.1.6. BGP neighbors submodule

```
  <CODE BEGINS> file "ietf-bgp-neighbor@2023-07-05.yang"
  submodule ietf-bgp-neighbor {
    yang-version 1.1;
    belongs-to ietf-bgp {
      prefix bgp;
    }

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

    import iana-bgp-types {
      prefix bt;
```

```
      reference
        "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
    }

    // Include the common submodule

    include ietf-bgp-common;
    include ietf-bgp-common-multiprotocol;
    include ietf-bgp-common-structure;

    // meta

    organization
      "IETF IDR Working Group";
    contact
      "WG Web:   <http://datatracker.ietf.org/wg/idr>
       WG List:  <idr@ietf.org>

       Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                Keyur Patel (keyur at arrcus.com),
                Susan Hares (shares at ndzh.com),
                Jeffrey Haas (jhaas at juniper.net).";

    description
      "This sub-module contains groupings that are specific to the
       neighbor context of the BGP module.

       Copyright (c) 2023 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 Revised 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 2023-07-05 {
```

```
      description
        "Initial Version";
      reference
        "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
    }

    grouping bgp-neighbor-afi-safi-list {
      description
        "List of address-families associated with the BGP neighbor";
      list afi-safi {
        key "name";
        description
          "AFI, SAFI configuration available for the neighbor or
           group";
        uses mp-afi-safi-config;
        leaf active {
          type boolean;
          config false;
          description
            "This value indicates whether a particular AFI-SAFI has
             been successfully negotiated with the peer. An AFI-SAFI
             may be enabled in the current running configuration, but
             a session restart may be required in order to negotiate
             the new capability.";
        }
        container prefixes {
          config false;
          description
            "Prefix counters for the AFI/SAFI in this BGP session";
          leaf received {
            type yang:zero-based-counter32;
            description
              "The number of prefixes received from the neighbor";
          }
          leaf sent {
            type yang:zero-based-counter32;
            description
              "The number of prefixes advertised to the neighbor";
          }
          leaf installed {
            type yang:gauge32;
            description
              "The number of advertised prefixes installed in the
               Loc-RIB";
          }
        }
        container graceful-restart {
          if-feature "bt:graceful-restart";
```

```
          description
            "Parameters relating to BGP graceful-restart";
          reference
            "RFC 4724: Graceful Restart Mechanism for BGP.";
          uses mp-afi-safi-graceful-restart-config;
          leaf received {
            type boolean;
            config false;
            description
              "This leaf indicates whether the neighbor advertised the
               ability to support graceful-restart for this AFI-SAFI";
            reference
              "RFC 4724: Graceful Restart Mechanism for BGP, Section
               3.";
          }
          leaf advertised {
            type boolean;
            config false;
            description
              "This leaf indicates whether the ability to support
               graceful-restart has been advertised to the peer";
            reference
              "RFC 4724: Graceful Restart Mechanism for BGP, Section
               3.";
          }
          leaf local-forwarding-state-preserved {
            type boolean;
            config false;
            description
              "This leaf indicates whether the local router has
               or would advertise the Forwarding State bit in its
               Graceful Restart capability for this AFI-SAFI.";
            reference
              "RFC 4724: Graceful Restart Mechanism for BGP, Section
               3.";
          }
          leaf forwarding-state-preserved {
            type boolean;
            config false;
            description
              "This leaf indicates whether the neighbor has advertised
               the Forwarding State bit in its Graceful Restart
               capability for this AFI-SAFI.";
            reference
              "RFC 4724: Graceful Restart Mechanism for BGP, Section
               3.";
          }
          leaf end-of-rib-received {
```

```
            type boolean;
            config false;
            description
              "This leaf indicates whether the neighbor has advertised
               the End-of-RIB marker for this AFI-SAFI.";
            reference
              "RFC 4724: Graceful Restart Mechanism for BGP, Section
               2.";
          }
        }
        uses mp-all-afi-safi-list-contents;
        uses bgp-neighbor-use-multiple-paths;
      }
    }
  }
  <CODE ENDS>
```

7.2. BGP notification module

```
  <CODE BEGINS> file "iana-bgp-notification@2023-07-05.yang"
  module iana-bgp-notification {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:iana-bgp-notification";
    prefix bn;

    // meta

    organization
      "IANA";
    contact
      "Internet Assigned Numbers Authority

       Postal: ICANN
               12025 Waterfront Drive, Suite 300
               Los Angeles, CA 90094-2536
               United States of America
       Tel:    +1 310 301 5800
       <mailto:iana@iana.org>

       Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                Keyur Patel (keyur at arrcus.com),
                Susan Hares (shares at ndzh.com),
                Jeffrey Haas (jhaas at juniper.net).";

    description
      "This YANG module is maintained by IANA and contains definitions
       of identities used to represent BGP NOTIFICATION Error
       code/Error Subcode tuples.  These values reflect the
```

```
        'BGP Error (Notification) Codes' and 'BGP Error Subcodes'
        registry at IANA.

        Copyright (c) 2023 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 Revised 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 2023-07-05 {
       description
         "Initial Version";
       reference
         "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
     }


     identity bgp-notification {
       description
         "Base identity for BGP NOTIFICATION state.";
     }

     identity message-header-error {
       base bgp-notification;
       description
         "All errors detected while processing the Message Header MUST
          be indicated by sending the NOTIFICATION message with the
          Error Code Message Header Error.

          The value of the 'Message Header Error' Error code is 1.";
       reference
         "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.1.";
     }
```

```
     identity message-header-unspecific {
       base message-header-error;
       description
         "If no appropriate Error Subcode is defined, then a zero
          (Unspecific) value is used for the Error Subcode field.

          The value of the 'Unspecific Error' subcode is 0.";
       reference
         "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.5.";
     }

     identity message-header-connection-not-synchronized {
       base message-header-error;
       description
         "If the Marker field of the message header is not as expected,
          then a synchronization error has occurred and the Error
          Subcode MUST be set to Connection Not Synchronized.

          The value of the 'Conection Not Synchronized Error' subcode is
          1.";
       reference
         "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.1.";
     }

     identity message-header-bad-message-length {
       base message-header-error;
       description
         "If at least one of the following is true:

          - if the Length field of the message header is less than 19 or
            greater than 4096, or

          - if the Length field of an OPEN message is less than the
            minimum length of the OPEN message, or

          - if the Length field of an UPDATE message is less than the
            minimum length of the UPDATE message, or

          - if the Length field of a KEEPALIVE message is not equal to
            19, or

          - if the Length field of a NOTIFICATION message is less than
            the minimum length of the NOTIFICATION message,

          then the Error Subcode MUST be set to Bad Message Length.  The
          Data field MUST contain the erroneous Length field.

          The value of the 'Bad Message Length' Error subcode is 2.";
```

```
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.1.";
    }

    identity message-header-bad-message-type {
      base message-header-error;
      description
        "If the Type field of the message header is not recognized,
         then the Error Subcode MUST be set to Bad Message Type.  The
         Data field MUST contain the erroneous Type field.

         The value of the 'Message Header Error' Error subcode is 3.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.1.";
    }

    identity open-message-error {
      base bgp-notification;
      description
        "All errors detected while processing the OPEN message MUST be
         indicated by sending the NOTIFICATION message with the Error
         Code 'OPEN Message Error'.

         The value of the 'OPEN Message Error' Error code is 2.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.2.";
    }

    identity open-message-unspecific {
      base open-message-error;
      description
        "From Section 4.5:
         If no appropriate Error Subcode is defined, then a zero
         (Unspecific) value is used for the Error Subcode field.

         The value of the 'Unspecific Error' subcode is 0.

         From Section 6.2:
         If one of the Optional Parameters in the OPEN message is
         recognized, but is malformed, then the Error Subcode MUST be
         set to 0 (Unspecific).";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.5.
         RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.2.";
    }

    identity open-unsupported-version-number {
      base open-message-error;
```

```
      description
        "If the version number in the Version field of the received
         OPEN message is not supported, then the Error Subcode MUST be
         set to Unsupported Version Number.  The Data field is a
         2-octet unsigned integer, which indicates the largest,
         locally-supported version number less than the version the
         remote BGP peer bid (as indicated in the received OPEN
         message), or if the smallest, locally-supported version number
         is greater than the version the remote BGP peer bid, then the
         smallest, locally-supported version number.

         The value of the 'Unsupported Version Number' Error subcode is
         1.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.2.";
    }

    identity open-bad-peer-as {
      base open-message-error;
      description
        "If the Autonomous System field of the OPEN message is
         unacceptable, then the Error Subcode MUST be set to Bad Peer
         AS.  The determination of acceptable Autonomous System numbers
         is outside the scope of this protocol.

         The value of the 'Bad Peer AS' Error subcode is 2.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.2.";
    }

    identity open-bad-bgp-id {
      base open-message-error;
      description
        "From RFC 4271, Section 6.2:
         If the BGP Identifier field of the OPEN message is
         syntactically incorrect, then the Error Subcode MUST be set to
         Bad BGP Identifier.  Syntactic correctness means that the BGP
         Identifier field represents a valid unicast IP host address.

         This was updated by RFC 6286:
         For a BGP speaker that supports the AS-wide Unique BGP
         Identifier, the OPEN message error handling related to the BGP
         Identifier is modified as follows:

         If the BGP Identifier field of the OPEN message is zero, or if
         it is the same as the BGP Identifier of the local BGP speaker
         and the message is from an internal peer, then the Error
         Subcode is set to 'Bad BGP Identifier'.
```

```
          The value of the 'Bad BGP Identifier' Error subcode is 3.";
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.2.
           RFC 6286: Autonomous-System-Wide Unique BGP Identifier for
           BGP-4., Section 2.2.";
      }

      identity open-unsupported-optional-parameter {
        base open-message-error;
        description
          "If one of the Optional Parameters in the OPEN message is not
          recognized, then the Error Subcode MUST be set to Unsupported
          Optional Parameters.

          The value of the 'Unsupported Optional Parameter' Error subcode
          is 4.";
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.2.";
      }

      identity open-unacceptable-hold-time {
        base open-message-error;
        description
          "If the Hold Time field of the OPEN message is unacceptable,
           then the Error Subcode MUST be set to Unacceptable Hold Time.
           An implementation MUST reject Hold Time values of one or two
           seconds.  An implementation MAY reject any proposed Hold Time.
           An implementation that accepts a Hold Time MUST use the
           negotiated value for the Hold Time.

           The value of the 'Unacceptable Hold Time' Error subcode is
           6.";
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.2.";
      }

      identity open-unsupported-capability {
        base open-message-error;
        description
          "If a BGP speaker that supports a certain capability determines
           that its peer doesn't support this capability, the speaker MAY
           send a NOTIFICATION message to the peer and terminate peering
           (see Section 'Extensions to Error Handling' for more details).
           For example, a BGP speaker may need to terminate peering if it
           established peering to exchange IPv6 routes and determines
           that its peer does not support Multiprotocol Extensions for
           BGP-4 [RFC4760].  The Error Subcode in the NOTIFICATION
           message is then set to Unsupported Capability.  The message
```

```
          MUST contain the capability or capabilities that cause the
          speaker to send the message.

          The value of the 'Unsupported Capability' Error subcode is
          7.";
      reference
        "RFC 5492: Capabilities Advertisement with BGP-4, Section 3.";
    }

    identity open-role-mismatch {
      base open-message-error;
      description
        "If the BGP Role Capability is advertised, and one is also
         received from the peer, the Roles MUST correspond to the
         relationships in Table 2. If the Roles do not correspond, the
         BGP speaker MUST reject the connection using the Role Mismatch
         Notification (code 2, subcode 11).";
      reference
        "RFC 9234: Route Leak Prevention and Detection Using Roles in
         UPDATE and OPEN Messages, Section 4.2.";
    }

    identity update-message-error {
      base bgp-notification;
      description
        "All errors detected while processing the UPDATE message MUST
         be indicated by sending the NOTIFICATION message with the
         Error Code UPDATE Message Error.

        The value of the 'UPDATE Message Error' Error code is 3.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.3.";
    }

    identity update-unspecific {
      base update-message-error;
      description
        "If no appropriate Error Subcode is defined, then a zero
         (Unspecific) value is used for the Error Subcode field.

         The value of the 'Unspecific Error' subcode is 0.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.5.";
    }

    identity update-malformed-attribute-list {
      base update-message-error;
      description
```

```
         "Error checking of an UPDATE message begins by examining the
          path attributes.  If the Withdrawn Routes Length or Total
          Attribute Length is too large (i.e., if Withdrawn Routes
          Length + Total Attribute Length + 23 exceeds the message
          Length), then the Error Subcode MUST be set to Malformed
          Attribute List.

          If any attribute appears more than once in the UPDATE message,
          then the Error Subcode MUST be set to Malformed Attribute
          List.

          The value of the 'Malformed Attribute List' Error subcode is
          1.";
       reference
         "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.3.";
     }

     identity update-unrecognized-well-known-attribute {
       base update-message-error;
       description
         "If any of the well-known mandatory attributes are not
          recognized, then the Error Subcode MUST be set to Unrecognized
          Well-known Attribute.  The Data field MUST contain the
          unrecognized attribute (type, length, and value).

          The value of the 'Unrecognized Well-known Attribute' Error
          subcode is 2.";
       reference
         "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.3.";
     }

     identity update-missing-well-known-attribute {
       base update-message-error;
       description
         "If any of the well-known mandatory attributes are not present,
          then the Error Subcode MUST be set to Missing Well-known
          Attribute.  The Data field MUST contain the Attribute Type
          Code of the missing, well-known attribute.

          The value of the 'Missing Well-known Attribute' Error subcode
          is 3.";
       reference
         "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.3.";
     }

     identity update-attribute-flags-error {
       base update-message-error;
       description
```

```
        "If any recognized attribute has Attribute Flags that conflict
         with the Attribute Type Code, then the Error Subcode MUST be
         set to Attribute Flags Error.  The Data field MUST contain the
         erroneous attribute (type, length, and value).

         The value of the 'Attribute Flags Error' Error subcode is 4.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.3.";
    }

    identity update-attribute-length-error {
      base update-message-error;
      description
        "If any recognized attribute has an Attribute Length that
         conflicts with the expected length (based on the attribute
         type code), then the Error Subcode MUST be set to Attribute
         Length Error.  The Data field MUST contain the erroneous
         attribute (type, length, and value).

         The value of the 'Attribute Length Error' Error subcode is
         5.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.3.";
    }

    identity update-invalid-origin-attribute {
      base update-message-error;
      description
        "If the ORIGIN attribute has an undefined value, then the Error
         Sub-code MUST be set to Invalid Origin Attribute.  The Data
         field MUST contain the unrecognized attribute (type, length,
         and value).

         The value of the 'Invalid ORIGIN Attribute' Error subcode is
         6.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.3.";
    }

    identity update-invalid-next-hop-attribute {
      base update-message-error;
      description
        "If the NEXT_HOP attribute field is syntactically incorrect,
         then the Error Subcode MUST be set to Invalid NEXT_HOP
         Attribute.  The Data field MUST contain the incorrect
         attribute (type, length, and value).  Syntactic correctness
         means that the NEXT_HOP attribute represents a valid IP host
         address.
```

```
          The value of the 'Invalid NEXT_HOP Attribute' Error subcode is
          8.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.3.";
    }

    identity open-optional-attribute-error {
      base update-message-error;
      description
        "If an optional attribute is recognized, then the value of this
         attribute MUST be checked.  If an error is detected, the
         attribute MUST be discarded, and the Error Subcode MUST be set
         to Optional Attribute Error.  The Data field MUST contain the
         attribute (type, length, and value).

         The value of the 'Optional Attribute Error' Error subcode is
         9.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.3.";
    }

    identity open-invalid-network-field {
      base update-message-error;
      description
        "The NLRI field in the UPDATE message is checked for syntactic
        validity.  If the field is syntactically incorrect, then the
        Error Subcode MUST be set to Invalid Network Field.

        The value of the 'Invalid Network Field' Error subcode is 10.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.3.";
    }

    identity open-malformed-as-path {
      base update-message-error;
      description
        "The AS_PATH attribute is checked for syntactic correctness.
         If the path is syntactically incorrect, then the Error Subcode
         MUST be set to Malformed AS_PATH.

         If the UPDATE message is received from an external peer, the
         local system MAY check whether the leftmost (with respect to
         the position of octets in the protocol message) AS in the
         AS_PATH attribute is equal to the autonomous system number of
         the peer that sent the message.  If the check determines this
         is not the case, the Error Subcode MUST be set to Malformed
         AS_PATH.
```

```
          The value of the 'Malformed AS_PATH' Error subcode is 11.";
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.3.";
      }

      identity hold-timer-expired-error {
        base bgp-notification;
        description
          "If a system does not receive successive KEEPALIVE, UPDATE,
           and/or NOTIFICATION messages within the period specified in
           the Hold Time field of the OPEN message, then the NOTIFICATION
           message with the Hold Timer Expired Error Code is sent and the
           BGP connection is closed.

           The value of the 'Hold Timer Expired' Error code is 4.";
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.5.";
      }

      identity timer-expired-unspecific {
        base hold-timer-expired-error;
        description
          "If no appropriate Error Subcode is defined, then a zero
           (Unspecific) value is used for the Error Subcode field.

           The value of the 'Unspecific Error' subcode is 0.";
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.5.";
      }


      identity fsm-error {
        base bgp-notification;
        description
          "Any error detected by the BGP Finite State Machine (e.g.,
           receipt of an unexpected event) is indicated by sending the
           NOTIFICATION message with the Error Code Finite State Machine
           Error.

           The value of the 'Finite State Machine Error' Error code is
           5.";
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.6.
           RFC 6608: Subcodes for BGP Finite State Machine Error.";
      }

      identity fsm-error-unspecified {
        base fsm-error;
```

```
      description
        "If no appropriate Error Subcode is defined, then a zero
         (Unspecific) value is used for the Error Subcode field.

         The value of the 'Unspecific Error' subcode is 0.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.5.
         RFC 6608: Subcodes for BGP Finite State Machine Error, Section
         3.";
    }

    identity fsm-error-unexpected-in-opensent {
      base fsm-error;
      description
        "If a BGP speaker receives an unexpected message (e.g.,
         KEEPALIVE/ UPDATE/ROUTE-REFRESH message) on a session in
         OpenSent state, it MUST send to the neighbor a NOTIFICATION
         message with the Error Code Finite State Machine Error and the
         Error Subcode 'Receive Unexpected Message in OpenSent State'.
         The Data field is a 1-octet, unsigned integer that indicates
         the type of the unexpected message.

         The value of the 'Receive Unexpected Message in OpenSent
         State' Error subcode is 1.";
      reference
        "RFC 6608: Subcodes for BGP Finite State Machine Error, Section
         4.";
    }

    identity fsm-error-unexpected-in-openconfirm {
      base fsm-error;
      description
        "If a BGP speaker receives an unexpected message (e.g.,
         OPEN/UPDATE/ ROUTE-REFRESH message) on a session in
         OpenConfirm state, it MUST send a NOTIFICATION message with
         the Error Code Finite State Machine Error and the Error
         Subcode 'Receive Unexpected Message in OpenConfirm State' to
         the neighbor.  The Data field is a 1-octet, unsigned integer
         that indicates the type of the unexpected message.

         The value of the 'Receive Unexpected Message in OpenConfirm
         State' Error subcode is 2.";
      reference
        "RFC 6608: Subcodes for BGP Finite State Machine Error, Section
         4.";
    }

    identity fsm-error-unexpected-in-established {
```

```
      base fsm-error;
      description
        "If a BGP speaker receives an unexpected message (e.g., OPEN
         message) on a session in Established State, it MUST send to
         the neighbor a NOTIFICATION message with the Error Code Finite
         State Machine Error and the Error Subcode 'Receive Unexpected
         Message in Established State'.  The Data field is a 1-octet,
         unsigned integer that indicates the type of the unexpected
         message.

         The value of the 'Receive Unexpected Message in Established
         State' Error subcode is 3.";
      reference
        "RFC 6608: Subcodes for BGP Finite State Machine Error, Section
         4.";
    }

    identity cease {
      base bgp-notification;
      description
        "In the absence of any fatal errors (that are indicated in this
         section), a BGP peer MAY choose, at any given time, to close
         its BGP connection by sending the NOTIFICATION message with
         the Error Code Cease.  However, the Cease NOTIFICATION message
         MUST NOT be used when a fatal error indicated by this section
         does exist.

         The value of the 'Cease' Error code is 6.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 6.7.";
    }

    identity cease-max-prefixes {
      base cease;
      description
        "From RFC 4271, Section 6.7:
         A BGP speaker MAY support the ability to impose a
         locally-configured, upper bound on the number of address
         prefixes the speaker is willing to accept from a neighbor.
         When the upper bound is reached, the speaker, under control of
         local configuration, either (a) discards new address prefixes
         from the neighbor (while maintaining the BGP connection with
         the neighbor), or (b) terminates the BGP connection with the
         neighbor.  If the BGP speaker decides to terminate its BGP
         connection with a neighbor because the number of address
         prefixes received from the neighbor exceeds the
         locally-configured, upper bound, then the speaker MUST send
         the neighbor a NOTIFICATION message with the Error Code
```

```
          Cease.

          From RFC 4486:
          If a BGP speaker decides to terminate its peering with a
          neighbor because the number of address prefixes received from
          the neighbor exceeds a locally configured upper bound (as
          described in [BGP-4]), then the speaker MUST send to the
          neighbor a NOTIFICATION message with the Error Code Cease and
          the Error Subcode 'Maximum Number of Prefixes Reached'.  The
          message MAY optionally include the Address Family information
          [BGP-MP] and the upper bound in the 'Data' field, as shown in
          Figure 1, where the meaning and use of the <AFI, SAFI> tuple
          is the same as defined in [BGP-MP], Section 7.

          The value of the 'Maximum Number of Prefixes Reached' Error
          subcode is 1.";
        reference
          "RFC 4486: Subcodes for BGP Cease Notification Message, Section
           4.";
      }

      identity cease-admin-shutdown {
        base cease;
        description
          "If a BGP speaker decides to administratively shut down its
           peering with a neighbor, then the speaker SHOULD send a
           NOTIFICATION message with the Error Code Cease and the Error
           Subcode 'Administrative Shutdown'.

           The value of the 'Administrative Shutdown' Error subcode is
           2.";
        reference
          "RFC 4486: Subcodes for BGP Cease Notification Message, Section
           4.";
      }

      identity cease-peer-deconfigured {
        base cease;
        description
          "If a BGP speaker decides to de-configure a peer, then the
           speaker SHOULD send a NOTIFICATION message with the Error Code
           Cease and the Error Subcode 'Peer De-configured'.

           The value of the 'Peer De-configured' Error subcode is 3.";
        reference
          "RFC 4486: Subcodes for BGP Cease Notification Message, Section
           4.";
      }
```

```
     identity cease-admin-reset {
       base cease;
       description
         "If a BGP speaker decides to administratively reset the peering
          with a neighbor, then the speaker SHOULD send a NOTIFICATION
          message with the Error Code Cease and the Error Subcode
          'Administrative Reset'.

          The value of the 'Administrative Reset' Error subcode is 4.";
        reference
         "RFC 4486: Subcodes for BGP Cease Notification Message, Section
          4.";
     }

     identity cease-connection-rejected {
       base cease;
       description
         "If a BGP speaker decides to disallow a BGP connection (e.g.,
          the peer is not configured locally) after the speaker accepts
          a transport protocol connection, then the BGP speaker SHOULD
          send a NOTIFICATION message with the Error Code Cease and the
          Error Subcode 'Connection Rejected'.

          The value of the 'Connection Rejected' Error subcode is 5.";
       reference
         "RFC 4486: Subcodes for BGP Cease Notification Message, Section
          4.";
     }

     identity cease-other-configuration-change {
       base cease;
       description
         "If a BGP speaker decides to administratively reset the peering
          with a neighbor due to a configuration change other than the
          ones described above, then the speaker SHOULD send a
          NOTIFICATION message with the Error Code Cease and the Error
          Subcode 'Other Configuration Change'.

          The value of the 'Other Configuration Change' Error subcode is
          6.";
       reference
         "RFC 4486: Subcodes for BGP Cease Notification Message, Section
          4.";
     }

     identity cease-connection-collision {
       base cease;
       description
```

```
        "If a BGP speaker decides to send a NOTIFICATION message with
         the Error Code Cease as a result of the collision resolution
         procedure (as described in [BGP-4]), then the subcode SHOULD
         be set to 'Connection Collision Resolution'.

         The value of the 'Connection Collision Resolution' Error
         subcode is 7.";
      reference
        "RFC 4486: Subcodes for BGP Cease Notification Message, Section
         4.";
    }

    identity cease-out-of-resources {
      base cease;
      description
        "If a BGP speaker runs out of resources (e.g., memory) and
         decides to reset a session, then the speaker MAY send a
         NOTIFICATION message with the Error Code Cease and the Error
         Subcode 'Out of Resources'.

         The value of the 'Out of Resources' Error subcode is 8.";
      reference
        "RFC 4486: Subcodes for BGP Cease Notification Message, Section
         4.";
    }

    identity cease-hard-reset {
      base cease;
      description
        "[RFC 8538] defines a new subcode for BGP Cease NOTIFICATION
         messages, called the Hard Reset subcode.  The value of this
         subcode is [9].  In this document, a BGP Cease NOTIFICATION
         message with the Hard Reset subcode is referred to as a 'Hard
         Reset message' or simply as a 'Hard Reset'.

         The value of the 'Hard Reset' Error subcode is 9.";
      reference
        "RFC 8538: Notification Message Support for BGP Graceful
         Restart, Section 3.";
    }

    identity cease-bfd-down {
      base cease;
      description
        "When a BGP session is terminated due to a BFD session going
         into the Down state, the BGP Speaker SHOULD send a
         NOTIFICATION message with the Error Code Cease and the Error
         Subcode 'BFD Down'.
```

```
          The value of the 'BFD Down' Error subocde is 10.";
        reference
          "draft-ietf-idr-bfd-subcode-05: A BGP Cease Notification
          Subcode For Bidirectional Forwarding Detection (BFD), Section
          2.";
      }

      identity route-refresh-message-error {
        base bgp-notification;
        description
          "The ROUTE-REFRESH Message Error code was defined in RFC 7313.

          The value of the 'ROUTE-REFRESH Message Error' Error code is
          7.";
        reference
          "RFC 7313: Enhanced Route Refresh Capability for BGP-4, Section
           5.";
      }

      identity route-refresh-reserved {
        base route-refresh-message-error;
        description
          "If no appropriate Error Subcode is defined, then a zero
           (Unspecific) value is used for the Error Subcode field.

           The value of the 'Unspecific Error' subcode is 0.";
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.5.
           RFC 7313: Enhanced Route Refresh Capability for BGP-4, Section
           6.";
      }

      identity route-refresh-invalid-message-length {
        base route-refresh-message-error;
        description
          "If the length, excluding the fixed-size message header, of the
           received ROUTE-REFRESH message with Message Subtype 1 and 2 is
           not 4, then the BGP speaker MUST send a NOTIFICATION message
           with the Error Code of 'ROUTE-REFRESH Message Error' and the
           subcode of 'Invalid Message Length'.  The Data field of the
           NOTIFICATION message MUST contain the complete ROUTE-REFRESH
           message.

           The value of the 'Invalid Message Length' Error subcode is
           1.";
        reference
          "RFC 7313: Enhanced Route Refresh Capability for BGP-4, Section
           5.";
```

```
    }
  }
  <CODE ENDS>
```

7.3. BGP types module

```
  <CODE BEGINS> file "iana-bgp-types@2023-07-05.yang"
  module iana-bgp-types {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:iana-bgp-types";
    prefix bt;

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

    // meta

    organization
      "IANA";
    contact
      "Internet Assigned Numbers Authority

       Postal: ICANN
               12025 Waterfront Drive, Suite 300
               Los Angeles, CA 90094-2536
               United States of America
       Tel:    +1 310 301 5800
       <mailto:iana@iana.org>

       Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                Keyur Patel (keyur at arrcus.com),
                Susan Hares (shares at ndzh.com),
                Jeffrey Haas (jhaas at juniper.net).";

    description
      "This module contains general data definitions for use in BGP.
       It can be imported by modules that make use of BGP attributes.

       Copyright (c) 2023 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 Revised 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 2023-07-05 {
       description
         "Initial Version";
       reference
         "RFC XXXX: YANG module for Border Gateway Protocol (BGP-4).";
     }

     /*
      * Features.
      */

     feature add-paths {
       description
         "Advertisement of multiple paths for the same address prefix
          without the new paths implicitly replacing any previous
          ones.";
       reference
         "RFC 7911: Advertisement of Multiple Paths in BGP.";
     }

     feature bfd {
       description
         "Support for BFD detection of BGP neighbor reachability.";
       reference
         "RFC 5880: Bidirectional Forward Detection (BFD),
          RFC 5881: Bidirectional Forward Detection for IPv4 and IPv6
                    (Single Hop),
          RFC 5883: Bidirectional Forwarding Detection (BFD) for
                    Multihop Paths.";
     }
```

```
     feature clear-neighbors {
       description
         "Clearing of BGP neighbors is supported.";
     }

     feature clear-routes {
       description
         "Clearing of BGP routes is supported.";
     }

     feature clear-statistics {
       description
         "Clearing of BGP statistics is supported.";
     }

     feature damping {
       description
         "Weighted route dampening is supported.";
     }

     feature graceful-restart {
       description
         "Graceful restart as defined in RFC 4724 is supported.";
     }

     feature route-refresh {
       description
         "Support for the BGP Route Refresh capability.";
       reference
         "RFC 2918: Route Refresh Capability for BGP-4.";
     }

     feature ttl-security {
       description
         "BGP Time To Live (TTL) security check support.";
       reference
         "RFC 5082: The Generalized TTL Security Mechanism (GTSM).";
     }

     /*
      * Identities.
      */

     /* BGP Origin Attribute Types. */

     typedef bgp-origin-attr-type {
       type enumeration {
         enum igp {
```

```
          description
            "Origin of the NLRI is internal";
        }
        enum egp {
          description
            "Origin of the NLRI is EGP";
        }
        enum incomplete {
          description
            "Origin of the NLRI is neither IGP or EGP";
        }
      }
      description
        "Type definition for standard BGP origin attribute";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Sec 4.3.";
    }

    /* BGP AS_PATH Segment Type Identities. */

    identity as-path-segment-type {
      description
        "Base AS Path Segment Type. In [BGP-4], the path segment type
         is a 1-octet field with the following values defined.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.3.";
    }

    identity as-set {
      base as-path-segment-type;
      description
        "Unordered set of autonomous systems that a route in the UPDATE
         message has traversed.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.3.";
    }

    identity as-sequence {
      base as-path-segment-type;
      description
        "Ordered set of autonomous systems that a route in the UPDATE
         message has traversed.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.3.";
    }

    identity as-confed-sequence {
      base as-path-segment-type;
```

```
      description
        "Ordered set of Member Autonomous Systems in the local
         confederation that the UPDATE message has traversed.";
      reference
        "RFC 5065: Autonomous System Configuration for BGP.";
    }

    identity as-confed-set {
      base as-path-segment-type;
      description
        "Unordered set of Member Autonomous Systems in the local
         confederation that the UPDATE message has traversed.";
      reference
        "RFC 5065: Autonomous System Configuration for BGP.";
    }

    /* BGP Peer-Types */

    typedef peer-type {
      type enumeration {
        enum internal {
          description
            "Internal (IBGP) peer";
        }
        enum external {
          description
            "External (EBGP) peer";
        }
        enum confederation-internal {
          description
            "Confederation Internal (IBGP) peer.";
        }
        enum confederation-external {
          description
            "Confederation External (EBGP) peer.";
        }
      }
      description
        "Labels a peer or peer group as explicitly internal,
         external, or the related confederation type.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Sec 1.1.
         RFC 5065: Autonomous System Configuration for BGP.";
    }

    /* BGP Capability Identities. */

    identity bgp-capability {
```

```
      description
        "Base identity for a BGP capability";
    }

    identity mp-bgp {
      base bgp-capability;
      description
        "Multi-protocol extensions to BGP";
      reference
        "RFC 4760: Multiprotocol Extentions for BGP-4.";
    }

    identity route-refresh {
      base bgp-capability;
      description
        "The BGP route-refresh functionality";
      reference
        "RFC 2918: Route Refresh Capability for BGP-4.";
    }

    identity asn32 {
      base bgp-capability;
      description
        "4-byte (32-bit) AS number functionality";
      reference
        "RFC6793: BGP Support for Four-Octet Autonomous System (AS)
                  Number Space.";
    }

    identity graceful-restart {
      if-feature "graceful-restart";
      base bgp-capability;
      description
        "Graceful restart functionality";
      reference
        "RFC 4724: Graceful Restart Mechanism for BGP.";
    }

    identity add-paths {
      if-feature "add-paths";
      base bgp-capability;
      description
        "Advertisement of multiple paths for the same address prefix
         without the new paths implicitly replacing any previous
         ones.";
      reference
        "RFC 7911: Advertisement of Multiple Paths in BGP.";
    }
```

```
/* BGP AFI-SAFI Type Identities. */

identity afi-safi-type {
  description
    "Base identity type for AFI,SAFI tuples for BGP-4";
  reference
    "RFC4760: Multiprotocol Extentions for BGP-4.";
}

identity ipv4-unicast {
  base afi-safi-type;
  description
    "IPv4 unicast (AFI,SAFI = 1,1)";
  reference
    "RFC4760: Multiprotocol Extentions for BGP-4.";
}

identity ipv4-labeled-unicast {
  base afi-safi-type;
  description
    "Labeled IPv4 unicast (AFI,SAFI = 1,4)";
  reference
    "RFC 8277: Using BGP to Bind MPLS Labels to Address Prefixes.";
}

identity ipv6-unicast {
  base afi-safi-type;
  description
    "IPv6 unicast (AFI,SAFI = 2,1)";
  reference
    "RFC4760: Multiprotocol Extentions for BGP-4.";
}

identity ipv6-labeled-unicast {
  base afi-safi-type;
  description
    "Labeled IPv6 unicast (AFI,SAFI = 2,4)";
  reference
    "RFC 8277: Using BGP to Bind MPLS Labels to Address Prefixes.";
}

identity l3vpn-ipv4-unicast {
  base afi-safi-type;
  description
    "Unicast IPv4 MPLS L3VPN (AFI,SAFI = 1,128)";
  reference
    "RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs).";
}
```

```
     identity l3vpn-ipv4-multicast {
       base afi-safi-type;
       description
         "Multicast IPv4 MPLS L3VPN (AFI,SAFI = 1,129)";
       reference
         "RFC 6514: BGP Encodings and Procedures for Multicast in
          MPLS/BGP IP VPNs.";
     }

     identity l3vpn-ipv6-unicast {
       base afi-safi-type;
       description
         "Unicast IPv6 MPLS L3VPN (AFI,SAFI = 2,128)";
       reference
         "RFC 4659: BGP-MPLS IP Virtual Private Network (VPN) Extension
          for IPv6 VPN.";
     }

     identity l3vpn-ipv6-multicast {
       base afi-safi-type;
       description
         "Multicast IPv6 MPLS L3VPN (AFI,SAFI = 2,129)";
       reference
         "RFC 6514: BGP Encodings and Procedures for Multicast in
          MPLS/BGP IP VPNs.";
     }

     identity l2vpn-evpn {
       base afi-safi-type;
       description
         "BGP MPLS Based Ethernet VPN (AFI,SAFI = 25,70)";
       reference
         "RFC 7432: BGP MPLS-Based Ethernet VPN.";
     }

     identity l2vpn-vpls {
       base afi-safi-type;
       description
         "BGP-signalled VPLS (AFI,SAFI = 25,65)";
       reference
         "RFC 4761: Virtual Private LAN Service (VPLS) Using BGP for
          Auto-Discovery and Signaling.";
     }

     /* BGP Remove Private AS Identities. */

     identity remove-private-as-option {
       description
```

```
        "Base identity for options for removing private autonomous
         system numbers from the AS_PATH attribute";
    }

    identity private-as-remove-all {
      base remove-private-as-option;
      description
        "Strip all private autonomous system numbers from the AS_PATH.
         This action is performed regardless of the other content of
         the AS_PATH attribute, and for all instances of private AS
         numbers within that attribute.";
    }

    identity private-as-replace-all {
      base remove-private-as-option;
      description
        "Replace all instances of private autonomous system numbers in
         the AS_PATH with the local BGP speaker's autonomous system
         number. This action is performed regardless of the other
         content of the AS_PATH attribute, and for all instances of
         private AS number within that attribute.";
    }

    /*
     * Typedefs.
     */

    typedef bgp-session-direction {
      type enumeration {
        enum inbound {
          description
            "Refers to all NLRI received from the BGP peer";
        }
        enum oubound {
          description
            "Refers to all NLRI advertised to the BGP peer";
        }
      }
      description
        "Type to describe the direction of NLRI transmission";
    }

    /* BGP Route Reflector Types. */

    typedef rr-cluster-id-type {
      type union {
        type uint32;
        type yang:dotted-quad;
```

```
      }
      description
        "Union type for route reflector cluster ids:
         option 1: 4-byte number
         option 2: IP address";
      reference
        "RFC 4456: BGP Route Reflection.";
    }

    /* BGP Graceful Restart Types. */

    typedef graceful-restart-time-type {
      type uint16 {
        range "0..4096";
      }
      units "seconds";
      description
        "This is the estimated time (in seconds) it will take for the
         BGP session to be re-established after a restart.  This can be
         used to speed up routing convergence by its peer in case that
         the BGP speaker does not come back after a restart.";
      reference
         "RFC 4724: Graceful Restart Mechanism for BGP, Section 3.";
    }

    typedef graceful-restart-flags {
      type bits {
        bit restart {
          position 0;
          description
            "The most significant bit is defined as the Restart
             State (R) bit, which can be used to avoid possible
             deadlock caused by waiting for the End-of-RIB marker
             when multiple BGP speakers peering with each other
             restart.  When set (value 1), this bit indicates
             that the BGP speaker has restarted, and its peer
             MUST NOT wait for the End-of-RIB marker from the
             speaker before advertising routing information to
             the speaker.";
          reference
            "RFC 4724: Graceful Restart Mechanism for BGP,
             Section 3.";
        }
        bit notification {
          position 1;
          description
            "The second most significant bit is defined in [RFC 8538]
             as the Graceful Notification ('N') bit.  It is used to
```

```
              indicate Graceful Restart support for BGP NOTIFICATION
              messages.  A BGP speaker indicates support for the
              procedures in this document by advertising a Graceful
              Restart Capability with its 'N' bit set (value 1).";
          reference
            "RFC 8538: Notification Message Support for BGP Graceful
              Restart, Section 2.";
        }
      }
      description
        "Restart Flags bits as defined for BGP Graceful Restart and its
         extensions.";
      reference
        "RFC 4724: Graceful Restart Mechanism for BGP,
         RFC 8538: Notification Message Support for BGP Graceful
         Restart.";
    }

    typedef graceful-restart-flags-for-afi {
      type bits {
        bit forwarding-preserved {
          position 0;
          description
            "The most significant bit is defined as the
             Forwarding State (F) bit, which can be used to
             indicate whether the forwarding state for routes
             that were advertised with the given AFI and SAFI
             has indeed been preserved during the previous BGP
             restart.  When set (value 1), the bit indicates
             that the forwarding state has been preserved.";
           reference
             "RFC 4724: Graceful Restart Mechanism for BGP,
              Section 3.";
        }
      }
      description
        "Flags for Address Family bits as defined for BGP Graceful
         Restart and its extensions.";
      reference
        "RFC 4724: Graceful Restart Mechanism for BGP.";
    }

    /* BGP Remove Private AS Types. */

    typedef remove-private-as-option {
      type identityref {
        base remove-private-as-option;
      }
```

```
      description
        "Set of options for configuring how private AS path numbers
         are removed from advertisements";
    }

  }
  <CODE ENDS>
```

7.4. BGP community types module

```
  <CODE BEGINS> file "iana-bgp-community-types@2023-07-05.yang"
  module iana-bgp-community-types {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:iana-bgp-community-types";
    prefix bct;

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

    // meta

    organization
      "IANA";
    contact
      "Internet Assigned Numbers Authority

       Postal: ICANN
               12025 Waterfront Drive, Suite 300
               Los Angeles, CA 90094-2536
               United States of America
       Tel:    +1 310 301 5800
       <mailto:iana@iana.org>

       Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                Keyur Patel (keyur at arrcus.com),
                Susan Hares (shares at ndzh.com),
                Jeffrey Haas (jhaas at juniper.net).";

    description
      "This module contains data definitions for BGP Communities in
       their various forms.  It can be imported by modules that make
       use of BGP attributes.
```

```
        This YANG module is maintained by IANA and reflects the
        'BGP Identities for Community' and 'BGP definitions for
        Community type' registries.

        Copyright (c) 2023 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 Revised 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 2023-07-05 {
       description
         "Initial Version";
       reference
         "RFC XXXX: YANG module for Border Gateway Protocol (BGP-4).";
     }

     /*
      * Features.
      */

     feature send-communities {
       description
         "Enable the propagation of communities.";
     }

     /*
      * Identities.
      */

     /* BGP Well-Known Standard (RFC 1997) Community Identities. */

     identity bgp-well-known-std-community {
       description
```

```
        "Base identity for reserved communities within the standard
         community space defined by RFC 1997. These communities must
         fall within the range 0xFFFF0000 to 0xFFFFFFFF";
      reference
        "RFC 1997: BGP Communities Attribute.";
    }

    identity no-export {
      base bgp-well-known-std-community;
      description
        "Do not export NLRI received carrying this community outside
         the bounds of this autonomous system, or this confederation
         (if the local autonomous system is a confederation member AS).
          This community has a value of 0xFFFFFF01.";
      reference
        "RFC 1997: BGP Communities Attribute.";
    }

    identity no-advertise {
      base bgp-well-known-std-community;
      description
        "All NLRI received carrying this community must not be
         advertised to other BGP peers. This community has a value of
         0xFFFFFF02.";
      reference
        "RFC 1997: BGP Communities Attribute.";
    }

    identity no-export-subconfed {
      base bgp-well-known-std-community;
      description
        "All NLRI received carrying this community must not be
         advertised to external BGP peers - including over
         confederation sub-AS boundaries. This community has a value of
         0xFFFFFF03.";
      reference
        "RFC 1997: BGP Communities Attribute.";
    }

    identity no-peer {
      base bgp-well-known-std-community;
      description
        "An autonomous system receiving NLRI tagged with this community
         is advised not to re-advertise the NLRI to external bilateral
         peer autonomous systems. An AS may also filter received NLRI
         from bilateral peer sessions when they are tagged with this
         community value. This community has a value of 0xFFFFFF04.";
      reference
```

```
        "RFC 3765: NOPEER Community for BGP.";
    }

    /* BGP Community Send Identities. */

    identity send-community-feature {
      description
        "Base identity to identify send-community feature.";
    }

    identity standard {
      base send-community-feature;
      description
        "Send standard communities.";
      reference
        "RFC 1997: BGP Communities Attribute.";
    }

    identity extended {
      base send-community-feature;
      description
        "Send extended communities.";
      reference
        "RFC 4360: BGP Extended Communities Attribute.";
    }

    identity large {
      base send-community-feature;
      description
        "Send large communities.";
      reference
        "RFC 8092: BGP Large Communities Attribute.";
    }

    /*
     * Typedefs.
     */

    /* BGP Community Types. */

    typedef bgp-community-regexp-type {
      type string;
      description
        "Type definition for communities specified as regular
         expression patterns.

         A compliant implementation of this type MUST accept a POSIX.2
         Extended Regular Expression excluding the following features:
```

```
          - character class expressions (Section 9.3.5)
          - collating symbols (Section 9.3.5)
          - equivalence classes (Section 9.3.5)
          - back-reference expressions (Section 9.3.6)

          Implementations MAY accept additional forms of regular
          expressions as long as the minimally compliant form documented
          above is accepted.";
      reference
        "IEEE Std 1003.1-2017: The Open Group Base Specifications Issue
         7, 2018 edition.";
    }

    typedef bgp-std-community-type {
      type union {
        type uint32;
        type string {
          pattern '([0-9]|[1-9][0-9]{1,3}|[1-5][0-9]{4}|'
                + '6[0-5][0-9]{3}|66[0-4][0-9]{2}|'
                + '665[0-2][0-9]|6653[0-5]):'
                + '([0-9]|[1-9][0-9]{1,3}|[1-5][0-9]{4}|'
                + '6[0-5][0-9]{3}|66[0-4][0-9]{2}|'
                + '665[0-2][0-9]|6653[0-5])';
        }
      }
      description
        "Type definition for standard community attributes.";
      reference
        "RFC 1997: BGP Communities Attribute.";
    }

    typedef bgp-well-known-community-type {
      type identityref {
        base bgp-well-known-std-community;
      }
      description
        "Type definition for well-known IETF community attribute
         values.";
      reference
        "IANA Border Gateway Protocol (BGP) Well Known Communities";
    }

    /* BGP Large Community Types. */

    typedef bgp-large-community-type {
      type string {
        // 4-octets global:4-octets local part-1:4-octets local part-2.
        pattern '(4[0-2][0-9][0-4][0-9][0-6][0-7][0-2][0-9][0-6]|'
```

```
               + '[1-3][0-9]{9}|[1-9]([0-9]{1,7})?[0-9]|[0-9]):'
               + '(4[0-2][0-9][0-4][0-9][0-6][0-7][0-2][0-9][0-6]|'
               + '[1-3][0-9]{9}|[1-9]([0-9]{1,7})?[0-9]|[0-9]):'
               + '(4[0-2][0-9][0-4][0-9][0-6][0-7][0-2][0-9][0-6]|'
               + '[1-3][0-9]{9}|[1-9]([0-9]{1,7})?[0-9]|[0-9])';
       }
       description
         "Type definition for a large BGP community";
       reference
         "RFC 8092: BGP Large Communities Attribute.";
     }

     /* IPv6 BGP Exteneded Community Types. */

     typedef bgp-ipv6-ext-community-type {
       type union {
         type string {
           pattern 'ipv6-route\-target:'
                + '((:|[0-9a-fA-F]{0,4}):)([0-9a-fA-F]{0,4}:){0,5}'
                + '((([0-9a-fA-F]{0,4}:)?(:|[0-9a-fA-F]{0,4}))|'
                + '(((25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9]?[0-9])\.){3}'
                + '(25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9]?[0-9])))'
                + ':'
                + '(6553[0-5]|655[0-2][0-9]|65[0-4][0-9]{2}|'
                + '6[0-4][0-9]{3}|'
                + '[1-5][0-9]{4}|[1-9][0-9]{0,3}|0)';
            pattern 'ipv6-route\-target:'
                + '((([^:]+:){6}(([^:]+:[^:]+)|(.*\..*)))|'
                + '((([^:]+:)*[^:]+)?::(([^:]+:)*[^:]+)?))'
                + ':'
                + '(6553[0-5]|655[0-2][0-9]|65[0-4][0-9]{2}|'
                + '6[0-4][0-9]{3}|'
                + '[1-5][0-9]{4}|[1-9][0-9]{0,3}|0)';
           /*
            * description
            *   "Type 0x00, Sub-Type 0x02: IPv6 Route-Target
            *    ipv6-route-target:(IPv6 address):(local-admin)
            *    16 octets global administrator and 2 octets local
            *    administrator.
            *
            *    Note: The patterns above are excerpted from RFC 8294.";
            * reference
            *   "RFC 5701: IPv6 Address Specific BGP Extended Community
            *    Attribute, Section 3.";
            */
         }

         type string {
```

```
          pattern 'ipv6-route\-origin:'
               + '((:|[0-9a-fA-F]{0,4}):)([0-9a-fA-F]{0,4}:){0,5}'
               + '((([0-9a-fA-F]{0,4}:)?(:|[0-9a-fA-F]{0,4}))|'
               + '(((25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9]?[0-9])\.){3}'
               + '(25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9]?[0-9])))'
               + ':'
               + '(6553[0-5]|655[0-2][0-9]|65[0-4][0-9]{2}|'
               + '6[0-4][0-9]{3}|'
               + '[1-5][0-9]{4}|[1-9][0-9]{0,3}|0)';
           pattern 'ipv6-route\-origin:'
               + '(([^:]+:){6}(([^:]+:[^:]+)|(.*\..*)))|'
               + '((([^:]+:)*[^:]+)?::(([^:]+:)*[^:]+)?))'
               + ':'
               + '(6553[0-5]|655[0-2][0-9]|65[0-4][0-9]{2}|'
               + '6[0-4][0-9]{3}|'
               + '[1-5][0-9]{4}|[1-9][0-9]{0,3}|0)';
          /*
           * description
           *   "Type 0x00, Sub-Type 0x03: IPv6 Route-Origin
           *    ipv6-route-origin:(IPv6 address):(local-admin)
           *    16 octets global administrator and 2 octets local
           *    administrator.
           *
           *    Note: The patterns above are excerpted from RFC 8294.";
           * reference
           *   "RFC 5701: IPv6 Address Specific BGP Extended Community
           *    Attribute, Section 3.";
           */
        }

        type string {
          // raw with 20 octets
          pattern 'ipv6-raw:'
               + '([0-9A-Fa-f][0-9A-Fa-f]:){19}'
               + '[0-9A-Fa-f][0-9A-Fa-f]';
        }
      }
      description
        "Type definition for IPv6 extended community attributes.
         It includes a way to specify a 'raw' string that
         is followed by 20 bytes of octet string to support
         new and experimental type definitions.";
      reference
        "RFC 5701: IPv6 Address Specific BGP Extended Community
         Attribute, Section 3.";
    }

    /* BGP Extended Community Types. */
```

```
     typedef bgp-ext-community-type {
       type union {
         type string {
           pattern 'route\-target:(6[0-5][0-5][0-3][0-5]|'
                 + '[1-5][0-9]{4}|[1-9][0-9]{1,4}|[0-9]):'
                 + '(4[0-2][0-9][0-4][0-9][0-6][0-7][0-2][0-9][0-6]|'
                 + '[1-3][0-9]{9}|[1-9]([0-9]{1,7})?[0-9]|[0-9])';
           /*
            * description
            *   "Type 0x00, Sub-Type 0x02: Route-Target
            *    route-target:(ASN):(local-admin)
            *    2 octets global administrator and 4 octets local
            *    administrator.";
            * reference
            *   "RFC 4360: BGP Extended Communities Attribute,
            *    Section 4.";
            */
         }

         type string {
           pattern 'route\-target:'
                 + '(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|'
                 + '25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9][0-9]|'
                 + '2[0-4][0-9]|25[0-5]):'
                 + '(6[0-5][0-5][0-3][0-5]|[1-5][0-9]{4}|'
                 + '[1-9][0-9]{1,4}|[0-9])';
           /*
            * description
            *   "Type 0x01, Sub-Type 0x02: Route-Target
            *    route-target:(IPv4):(local-admin)
            *    4 octets IP address global administrator and 2 octets
            *    local administrator.";
            * reference
            *   "RFC 4360: BGP Extended Communities Attribute,
            *    Section 4.";
            */
         }

         type string {
           pattern 'route\-target:'
                 + '(4[0-2][0-9][0-4][0-9][0-6][0-7][0-2][0-9][0-6]|'
                 + '[1-3][0-9]{9}|[1-9]([0-9]{1,7})?[0-9]|[0-9])'
                 + '(6[0-5][0-5][0-3][0-5]|[1-5][0-9]{4}|'
                 + '[1-9][0-9]{1,4}|[0-9])';
           /*
            * description
            *   "Type 0x02, Sub-Type 0x02: Route-Target
            *    route-target:(ASN):(local-admin)
```

```
           *    4 octets global administrator and 2 octets local
           *    administrator.";
           * reference
           *   "RFC 5668: 4-Octet AS Specific BGP Extended Community,
           *    Section 4.";
           */
        }

        type string {
          pattern 'route\-origin:(6[0-5][0-5][0-3][0-5]|'
                + '[1-5][0-9]{4}|[1-9][0-9]{1,4}|[0-9]):'
                + '(4[0-2][0-9][0-4][0-9][0-6][0-7][0-2][0-9][0-6]|'
                + '[1-3][0-9]{9}|[1-9]([0-9]{1,7})?[0-9]|[0-9])';
          /*
           * description
           *   "Type 0x01, Sub-Type 0x03: Route-Origin
           *    route-target:(ASN):(local-admin)
           *    2 octets global administrator and 4 octets local
           *    administrator.";
           * reference
           *   "RFC 4360: BGP Extended Communities Attribute,
           *    Section 5.";
           */
        }

        type string {
          pattern 'route\-origin:'
                + '(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|'
                + '25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9][0-9]|'
                + '2[0-4][0-9]|25[0-5]):'
                + '(6[0-5][0-5][0-3][0-5]|[1-5][0-9]{4}|'
                + '[1-9][0-9]{1,4}|[0-9])';
          /*
           * description
           *   "Type 0x01, Sub-Type 0x03: Route-Target
           *    route-target:(IPv4):(local-admin)
           *    4 octets IP address global administrator and 2 octets
           *    local administrator.";
           * reference
           *   "RFC 4360: BGP Extended Communities Attribute,
           *    Section 5.";
           */
        }

        type string {
          pattern 'route\-origin:'
                + '(4[0-2][0-9][0-4][0-9][0-6][0-7][0-2][0-9][0-6]|'
                + '[1-3][0-9]{9}|[1-9]([0-9]{1,7})?[0-9]|[0-9])'
```

```
                 + '(6[0-5][0-5][0-3][0-5]|[1-5][0-9]{4}|'
                 + '[1-9][0-9]{1,4}|[0-9])';
           /*
            * description
            *   "Type 0x02, Sub-Type 0x03: Route-Origin
            *    route-origin:(ASN):(local-admin)
            *    4 octets global administrator and 2 octets local
            *    administrator.";
            * reference
            *   "RFC 5668: 4-Octet AS Specific BGP Extended Community,
            *    Section 4.";
            */
         }

         type string {
           // raw with 8 octets
           pattern 'raw:'
                + '([0-9A-Fa-f][0-9A-Fa-f]:){7}'
                + '[0-9A-Fa-f][0-9A-Fa-f]';
         }
       }
       description
         "Type definition for extended community attributes.
          It includes a way to specify a 'raw' string that
          is followed by 8 bytes of octet string to support
          new and experimental type definitions.";
       reference
         "RFC 4360: BGP Extended Communities Attribute.";
     }
   }
   <CODE ENDS>
```

7.5. BGP policy module

```
   <CODE BEGINS> file "ietf-bgp-policy@2023-07-05.yang"
   module ietf-bgp-policy {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-bgp-policy";
     prefix bp;

     // import some basic types

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

```
      prefix rt-pol;
      reference
        "RFC 9067: A YANG Data Model for Routing Policy";
    }
    import iana-bgp-types {
      prefix bt;
      reference
        "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
    }
    import iana-bgp-community-types {
      prefix bct;
      reference
        "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
    }

    organization
      "IETF IDR Working Group";
    contact
      "WG Web:   <http://datatracker.ietf.org/wg/idr>
       WG List:  <idr@ietf.org>

       Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                Keyur Patel (keyur at arrcus.com),
                Susan Hares (shares at ndzh.com),
                Jeffrey Haas (jhaas at juniper.net).";

    description
      "This module contains data definitions for BGP routing policy.
       It augments the base routing-policy module with BGP-specific
       options for conditions and actions.

       Copyright (c) 2023 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 Revised 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 2023-07-05 {
       description
         "Initial Version";
       reference
         "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
     }

     /*
      * typedef statements
      */

     typedef bgp-set-community-option-type {
       type enumeration {
         enum add {
           description
             "Add the specified communities to the existing
              community attribute.";
         }
         enum remove {
           description
             "Remove the specified communities from the
              existing community attribute.";
         }
         enum replace {
           description
             "Replace the existing community attribute with
              the specified communities. If an empty set is
              specified, this removes the community attribute
              from the route.";
         }
       }
       description
         "Type definition for options when setting the community
          attribute in a policy action.";
     }

     typedef bgp-next-hop-type {
       type union {
         type inet:ip-address-no-zone;
         type enumeration {
           enum self {
             description
               "Special designation for local router's own
                address, i.e., next-hop-self.";
           }
```

```
        }
      }
      description
        "Type definition for specifying next-hop in policy actions.";
    }

    typedef bgp-set-med-type {
      type union {
        type string {
          pattern '[+-]([0-9]{1,8}|[0-3][0-9]{1,9}|4[0-1][0-9]{1,8}|'
                + '428[0-9]{1,7}|429[0-3][0-9]{1,6}|42948[0-9]{1,5}|'
                + '42949[0-5][0-9]{1,4}|429496[0-6][0-9]{1,3}|'
                + '4294971[0-9]{1,2}|42949728[0-9]|42949729[0-5])';
        }
        type enumeration {
          enum igp {
            description
              "Set the MED value to the IGP cost toward the
               next hop for the route.";
          }
          enum med-plus-igp {
            description
              "Before comparing MED values for path selection, adds to
               the MED the cost of the IGP route to the BGP next-hop
               destination.

               This option replaces the MED value for the router,
               but does not affect the IGP metric comparison. As a
               result, when multiple routes have the same value
               after the MED-plus-IPG comparison, and route selection
               continues, the IGP route metric is also compared, even
               though it was added to the MED value and compared
               earlier in the selection process.

               Useful when the downstream AS requires the complete
               cost of a certain route that is received across
               multiple ASs.";
          }
        }
        type uint32;
      }
      description
        "Type definition for specifying how the BGP MED can
         be set in BGP policy actions. The three choices are to set
         the MED directly, increment/decrement using +/- notation,
         and setting it to the IGP cost (predefined value).";
    }
```

```
     grouping equality-operator {
       description
         "Grouping used for common equality operations in BGP policy.";
       choice operation {
         case eq {
           leaf eq {
             type empty;
             description
               "Check to see if the value is equal.";
           }
         }

         case lt-or-eq {
           leaf lt-or-eq {
             type empty;
             description
               "Check to see if the value is less than or equal.";
           }
         }

         case gt-or-eq {
           leaf gt-or-eq {
             type empty;
             description
               "Check to see if the value is greater than or
                equal.";
           }
         }
         description
           "Choice of operations on the value";
       }
     }

     // Identities

     // augment statements

     augment "/rt-pol:routing-policy/rt-pol:defined-sets" {
       description
         "Adds BGP defined sets container to routing policy model.";
       container bgp-defined-sets {
         description
           "BGP-related set definitions for policy match conditions.";
         container as-path-sets {
           description
             "Enclosing container for list of define AS path sets.";
           list as-path-set {
             key "name";
```

```
            description
              "List of defined AS path sets.";
            leaf name {
              type string;
              description
                "Name of the AS path set -- this is used to reference
                 the set in match conditions.";
            }
            leaf-list member {
              type string;
              description
                "AS path regular expression using BGP YANG AS_PATH
                 regular expression syntax.  If any of the regular
                 expressions in the lists are matched, the as-path-set
                 is considered matched.";
              reference
                "RFC XXXX: YANG Model for Border Gateway Protocol
                 (BGP-4), Appendix F.3.";
            }
          }
        }

        container community-sets {
          description
            "Enclosing container for list of defined BGP community
             sets.";
          list community-set {
            key "name";
            description
              "List of defined BGP community sets.";
            leaf name {
              type string;
              description
                "Name / label of the community set -- this is used to
                 reference the set in match conditions.";
            }
            leaf-list member {
              type union {
                type bct:bgp-well-known-community-type;
                type bct:bgp-std-community-type;
                type bct:bgp-community-regexp-type;
              }
              description
                "Members of the community set";
            }
          }
        }
```

```
        container ext-community-sets {
          description
            "Enclosing container for list of extended BGP community
             sets";
          list ext-community-set {
            key "name";
            description
              "List of defined extended BGP community sets";
            leaf name {
              type string;
              description
                "Name / label of the extended community set -- this is
                 used to reference the set in match conditions";
            }
            leaf-list member {
              type union {
                type bct:bgp-ext-community-type;
                type bct:bgp-community-regexp-type;
              }
              description
                "Members of the extended community set.";
            }
          }
        }

        container ipv6-ext-community-sets {
          description
            "Enclosing container for list of extended IPv6 BGP
             community sets";
          list ipv6-ext-community-set {
            key "name";
            description
              "List of defined IPv6 extended BGP community sets";
            leaf name {
              type string;
              description
                "Name / label of the IPv6 extended community set --
                 this is used to reference the set in match
                 conditions";
            }
            leaf-list member {
              type union {
                type bct:bgp-ipv6-ext-community-type;
                type bct:bgp-community-regexp-type;
              }
              description
                "Members of the IPv6 extended community set.";
            }
```

```
          }
        }

        container large-community-sets {
          description
            "Enclosing container for list of large BGP community
             sets";
          list large-community-set {
            key "name";
            description
              "List of defined large BGP community sets";
            leaf name {
              type string;
              description
                "Name / label of the large community set -- this is
                 used to reference the set in match conditions";
            }
            leaf-list member {
              type union {
                type bct:bgp-large-community-type;
                type bct:bgp-community-regexp-type;
              }
              description
                "Members of the large community set.";
            }
          }
        }

        container next-hop-sets {
          description
            "Definition of a list of IPv4 or IPv6 next-hops which can
             be matched in a routing policy.";

          list next-hop-set {
            key "name";
            description
              "List of defined next-hop sets for use in policies.";

            leaf name {
              type string;
              description
                "Name of the next-hop set.";
            }
            leaf-list next-hop {
              type bgp-next-hop-type;
              description
                "List of IP addresses in the next-hop set.";
            }
```

```
          }
        }
      }
    }

    augment "/rt-pol:routing-policy/rt-pol:policy-definitions/" +
            "rt-pol:policy-definition/rt-pol:statements/" +
            "rt-pol:statement/rt-pol:conditions" {
      description
        "BGP policy conditions added to routing policy module.";

      container bgp-conditions {
        description
          "Top-level container for BGP specific policy conditions.";

        container local-pref {
          description
            "Value and comparison operations for conditions based on
             the value of the BGP LOCAL_PREF.";
          reference
            "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
             5.1.5.";

          leaf value {
            type uint32;
            description
              "BGP LOCAL_PREF value for comparison to the entry in the
               BGP route.";
          }

          uses equality-operator;
        }

        container med {
          description
            "Value and comparison operations for conditions based on
             the value of the BGP MULTI_EXIT_DISC (MED)";
          reference
            "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
             5.1.4.";

          leaf value {
            type uint32;
            description
              "BGP MED value for comparison to the entry in the BGP
               route.";
          }
```

```
          uses equality-operator;
        }

        leaf origin-eq {
          type bt:bgp-origin-attr-type;
          description
            "Condition to check if the route origin is equal to the
             specified value.";
        }

        container match-afi-safi {
          description
            "Match an address family according to the logic defined in
             the match-set-options leaf.";
          leaf-list afi-safi-in {
            type identityref {
              base bt:afi-safi-type;
            }
            description
              "List of address families which the NLRI may be within.";
          }
          uses rt-pol:match-set-options-restricted-group;
        }

        container match-neighbor {
          description
            "Match a neighbor according to the logic defined in the
             match-set-options leaf.";

          leaf-list neighbor-eq {
            type inet:ip-address;
            description
              "List of neighbor addresses to check for in the ingress
               direction.";
          }
          uses rt-pol:match-set-options-restricted-group;
        }

        leaf route-type {
          type enumeration {
            enum internal {
              description
                "route type is internal.";
            }
            enum external {
              description
                "route type is external.";
            }
```

```
          }
          description
            "Condition to check the route type in the route update.";
        }

        container community-count {
          description
            "Value and comparison operations for conditions based on
             the number of communities in the route update.";

          leaf community-count {
            type uint32;
            description
              "Value for the number of communities in the route
               update.";
          }

          uses equality-operator;
        }

        container as-path-length {
          description
            "Value and comparison operations for conditions based on
             the length of the AS path in the route update.

             The as-path-length SHALL be calculated and SHALL follow
             RFC 4271 rules.";
          reference
            "RFC 4271: A Border Gateway Protocol 4 (BGP-4).";

          leaf as-path-length {
            type uint32;
            description
              "Value of the AS path length in the route update.";
          }

          uses equality-operator;
        }

        container match-community-set {
          description
            "Top-level container for match conditions on communities.
             Match a referenced community-set according to the logic
             defined in the match-set-options leaf.";
          leaf community-set {
            type leafref {
              path "/rt-pol:routing-policy/rt-pol:defined-sets/"
                 + "bgp-defined-sets/community-sets/"
```

```
                 + "community-set/name";
             }
             description
               "References a defined community set.";
           }
           uses rt-pol:match-set-options-group;
         }

         container match-ext-community-set {
           description
             "Match a referenced extended community-set according to the
              logic defined in the match-set-options leaf.";
           leaf ext-community-set {
             type leafref {
               path "/rt-pol:routing-policy/rt-pol:defined-sets/"
                  + "bgp-defined-sets/ext-community-sets/"
                  + "ext-community-set/name";
             }
             description
               "References a defined extended community set.";
           }
           leaf ext-community-match-kind {
             type enumeration {
               enum ext-community {
                 description
                   "Perform the match against the ext-community RIB
                    attribute.";
               }
               enum ext-community-raw {
                 description
                   "Perform the match against the ext-community-raw RIB
                    attribute.";
               }
             }
             default ext-community;
             description
               "Extended communities may be matched by the ext-community
                RIB attribute, or the ext-community-raw RIB attribute.
                This leaf selects which leaf to perform the match
                operation against.";
           }
           uses rt-pol:match-set-options-group;
         }

         container match-ipv6-ext-community-set {
           description
             "Match a referenced IPv6 extended community-set according
              to the logic defined in the match-set-options leaf.";
```

```
          leaf ipv6-ext-community-set {
            type leafref {
              path "/rt-pol:routing-policy/rt-pol:defined-sets/"
                 + "bgp-defined-sets/ipv6-ext-community-sets/"
                 + "ipv6-ext-community-set/name";
            }
            description
              "References a defined IPv6 extended community set.";
          }
          leaf ipv6-ext-community-match-kind {
            type enumeration {
              enum ipv6-ext-community {
                description
                  "Perform the match against the IPv6 ext-community RIB
                   attribute.";
              }
              enum ipv6-ext-community-raw {
                description
                  "Perform the match against the IPv6 ext-community-raw
                   RIB attribute.";
              }
            }
            default ipv6-ext-community;
            description
              "IPv6 Extended communities may be matched by the
               ipv6-ext-community RIB attribute, or the
               ipv6-ext-community-raw RIB attribute.  This leaf selects
               which leaf to perform the match operation against.";
          }
          uses rt-pol:match-set-options-group;
        }

        container match-large-community-set {
          description
            "Match a referenced large community-set according to the
             logic defined in the match-set-options leaf.";
          leaf large-community-set {
            type leafref {
              path "/rt-pol:routing-policy/rt-pol:defined-sets/"
                 + "bgp-defined-sets/large-community-sets/"
                 + "large-community-set/name";
            }
            description
              "References a defined large community set.";
          }
          uses rt-pol:match-set-options-group;
        }
```

```
        container match-as-path-set {
          description
            "Match a referenced as-path set according to the logic
             defined in the match-set-options leaf.";
          leaf as-path-set {
            type leafref {
              path "/rt-pol:routing-policy/rt-pol:defined-sets/"
                 + "bgp-defined-sets/as-path-sets/"
                 + "as-path-set/name";
            }
            description
              "References a defined AS path set";
          }
          uses rt-pol:match-set-options-group;
        }

        container match-next-hop-set {
          description
            "Match a referenced next-hop set according to the logic
             defined in the match-set-options leaf.";
          leaf next-hop-set {
            type leafref {
              path "/rt-pol:routing-policy/rt-pol:defined-sets/"
                 + "bgp-defined-sets/next-hop-sets/"
                 + "next-hop-set/name";
            }
            description
              "Reference a defined next-hop set.";
          }
          uses rt-pol:match-set-options-restricted-group;
        }
      }
    }

    augment "/rt-pol:routing-policy/rt-pol:policy-definitions/" +
            "rt-pol:policy-definition/rt-pol:statements/" +
            "rt-pol:statement/rt-pol:actions" {
      description
        "BGP policy actions added to routing policy module.";
      container bgp-actions {
        description
          "Top-level container for BGP-specific actions";
        leaf set-route-origin {
          type bt:bgp-origin-attr-type;
          description
            "Set the origin attribute to the specified value";
        }
        leaf set-local-pref {
```

```
          type uint32;
          description
            "Set the local pref attribute on the route.";
        }
        leaf set-next-hop {
          type bgp-next-hop-type;
          description
            "Set the next-hop attribute in the route.";
        }
        leaf set-med {
          type bgp-set-med-type;
          description
            "Set the med metric attribute in the route.";
        }

        container set-as-path-prepend {
          description
            "Action to prepend local AS number to the AS-path a
             specified number of times";

          leaf repeat-n {
            type uint8 {
              range "1..max";
            }
            description
              "Number of times to prepend the AS number to the AS
               path. The value should be between 1 and the maximum
               supported by the implementation.";
          }
          leaf-list asn {
            type inet:as-number;
            description
              "AS number to prepend to the AS path.";
          }
        }

        container set-community {
          description
            "Action to set the community attributes of the route, along
             with options to modify how the community is modified.
             Communities may be set using an inline list OR
             reference to an existing defined set (not both).";

          leaf options {
            type bgp-set-community-option-type;
            description
              "Options for modifying the community attribute with
               the specified values.  These options apply to both
```

```
                 methods of setting the community attribute.";
           }

           choice method {
             description
               "Indicates the method used to specify the extended
                communities for the set-community action";
             case inline {
               leaf-list communities {
                 type union {
                   type bct:bgp-well-known-community-type;
                   type bct:bgp-std-community-type;
                 }
                 description
                   "Set the community values for the update inline with
                    a list.";
               }
             }

             case reference {
               leaf community-set-ref {
                 type leafref {
                   path "/rt-pol:routing-policy/rt-pol:defined-sets/"
                      + "bgp-defined-sets/"
                      + "community-sets/community-set/name";
                 }
                 description
                   "References a defined community set by name";
               }
             }
           }
         }

         container set-ext-community {
           description
             "Action to set the extended community attributes of the
              route, along with options to modify how the community is
              modified. Extended communities may be set using an inline
              list OR a reference to an existing defined set (but not
              both).";

           leaf options {
             type bgp-set-community-option-type;
             description
               "Options for modifying the community attribute with
                the specified values.  These options apply to both
                methods of setting the community attribute.";
           }
```

```
          choice method {
            description
              "Indicates the method used to specify the extended
               communities for the set-ext-community action";
            case inline {
              leaf-list communities {
                type bct:bgp-ext-community-type;
                description
                  "Set the extended community values for the update
                   inline with a list.";
              }
            }
            case reference {
              leaf ext-community-set-ref {
                type leafref {
                  path "/rt-pol:routing-policy/rt-pol:defined-sets/"
                     + "bgp-defined-sets/ext-community-sets/"
                     + "ext-community-set/name";
                }
                description
                  "References a defined extended community set by
                   name.";
              }
            }
          }
        }

        container set-ipv6-ext-community {
          description
            "Action to set the IPv6 extended community attributes of
             the route, along with options to modify how the community
             is modified. IPv6 extended communities may be set using an
             inline list OR a reference to an existing defined set (but
             not both).";

          leaf options {
            type bgp-set-community-option-type;
            description
              "Options for modifying the community attribute with
               the specified values.  These options apply to both
               methods of setting the community attribute.";
          }

          choice method {
            description
              "Indicates the method used to specify the IPv6 extended
               communities for the set-ipv6-ext-community action";
            case inline {
```

```
              leaf-list communities {
                type bct:bgp-ipv6-ext-community-type;
                description
                  "Set the IPv6 extended community values for the
                   update inline with a list.";
              }
            }
            case reference {
              leaf ipv6-ext-community-set-ref {
                type leafref {
                  path "/rt-pol:routing-policy/rt-pol:defined-sets/"
                     + "bgp-defined-sets/ipv6-ext-community-sets/"
                     + "ipv6-ext-community-set/name";
                }
                description
                  "References a defined IPv6 extended community set by
                   name.";
              }
            }
          }
        }

        container set-large-community {
          description
            "Action to set the large community attributes of the
             route, along with options to modify how the community is
             modified. Large communities may be set using an inline
             list OR a reference to an existing defined set (but not
             both).";

          leaf options {
            type bgp-set-community-option-type;
            description
              "Options for modifying the community attribute with
               the specified values. These options apply to both
               methods of setting the community attribute.";
          }

          choice method {
            description
              "Indicates the method used to specify the large
               communities for the set-large-community action";
            case inline {
              leaf-list communities {
                type bct:bgp-large-community-type;
                description
                  "Set the large community values for the update
                   inline with a list.";
```

```
              }
            }
            case reference {
              leaf large-community-set-ref {
                type leafref {
                  path "/rt-pol:routing-policy/rt-pol:defined-sets/"
                     + "bgp-defined-sets/large-community-sets/"
                     + "large-community-set/name";
                }
                description
                  "References a defined extended community set by
                   name.";
              }
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

7.6. RIB modules and submodules

7.6.1. BGP RIB types module

```
  <CODE BEGINS> file "iana-bgp-rib-types@2023-07-05.yang"
  module iana-bgp-rib-types {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:iana-bgp-rib-types";
    prefix brt;

    organization
      "IANA";
    contact
      "Internet Assigned Numbers Authority

       Postal: ICANN
               12025 Waterfront Drive, Suite 300
               Los Angeles, CA 90094-2536
               United States of America
       Tel:    +1 310 301 5800
       <mailto:iana@iana.org>

       Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                Keyur Patel (keyur at arrcus.com),
                Susan Hares (shares at ndzh.com),
                Jeffrey Haas (jhaas at juniper.net).";
```

```
     description
       "Defines identity and type definitions associated with
        the BGP RIB modules.

        Copyright (c) 2023 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 Revised 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 2023-07-05 {
       description
         "Initial Version";
       reference
         "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
     }

     identity ineligible-route-reason {
       description
         "Base identity for reason code for routes that are rejected as
          ineligible. Some derived entities are based on BMP v3.";
       reference
         "RFC 7854: BGP Monitoring Protocol.";
     }

     identity ineligible-cluster-loop {
       base ineligible-route-reason;
       description
         "Route was ineligible due to CLUSTER_LIST loop";
     }

     identity ineligible-as-loop {
       base ineligible-route-reason;
       description
```

```
        "Route was ineligible due to AS_PATH loop";
    }

    identity ineligible-originator {
      base ineligible-route-reason;
      description
        "Route was ineligible due to ORIGINATOR_ID. For example, update
         has local router as originator";
    }

    identity ineligible-confed {
      base ineligible-route-reason;
      description
        "Route was ineligible due to a loop in the AS_CONFED_SEQUENCE
         or AS_CONFED_SET attributes";
    }

    identity bgp-not-selected-bestpath {
      description
        "Base identity for indicating reason a route was was not
         selected by BGP route selection algorithm";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 9.1.";
    }

    identity local-pref-lower {
      base bgp-not-selected-bestpath;
      description
        "Route has a lower localpref attribute than current best path";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
         Section 9.1.2.";
    }

    identity as-path-longer {
      base bgp-not-selected-bestpath;
      description
        "Route has a longer AS path attribute than current best path";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
         Section 9.1.2.2 (a).";
    }

    identity origin-type-higher {
      base bgp-not-selected-bestpath;
      description
        "Route has a higher origin type, i.e., IGP origin is preferred
         over EGP or incomplete";
```

```
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
         Section 9.1.2.2 (b).";
    }

    identity med-higher {
      base bgp-not-selected-bestpath;
      description
        "Route has a higher MED, or metric, attribute than the current
         best path";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
         Section 9.1.2.2 (c).";
    }

    identity prefer-external {
      base bgp-not-selected-bestpath;
      description
        "Route source is via IBGP, rather than EGP.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
         Section 9.1.2.2 (d).";
    }

    identity nexthop-cost-higher {
      base bgp-not-selected-bestpath;
      description
        "Route has a higher interior cost to the next hop.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
         Section 9.1.2.2 (e).";
    }

    identity higher-router-id {
      base bgp-not-selected-bestpath;
      description
        "Route was sent by a peer with a higher BGP Identifier value.";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
         Section 9.1.2.2 (f).";
    }

    identity higher-peer-address {
      base bgp-not-selected-bestpath;
      description
        "Route was sent by a peer with a higher IP address";
      reference
        "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
```

```
          Section 9.1.2.2 (g).";
     }

     identity bgp-not-selected-policy {
       description
         "Base identity for reason code for routes that are rejected
          due to policy";
     }

     identity rejected-import-policy {
       base bgp-not-selected-policy;
       description
         "Route was rejected after applying import policies.";
     }
   }
   <CODE ENDS>
```

7.6.2. BGP RIB submodule

```
   <CODE BEGINS> file "ietf-bgp-rib@2023-07-05.yang"
   submodule ietf-bgp-rib {
     yang-version 1.1;
     belongs-to ietf-bgp {
       prefix bgp;
     }

     /*
      * Import and Include
      */

     import iana-bgp-types {
       prefix bt;
       reference
         "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
     }
     import ietf-inet-types {
       prefix inet;
       reference
         "RFC 6991: Common YANG Types.";
     }
     import ietf-yang-types {
       prefix yang;
       reference
         "RFC 6991: Common YANG Types.";
     }
     include ietf-bgp-rib-tables;

     // groupings of attributes in three categories:
```

```
     //  - shared across multiple routes
     //  - common to Loc-RIB and Adj-RIB, but not shared across routes
     //  - specific to Loc-RIB or Adj-RIB
     // groupings of annotations for each route or table
     include ietf-bgp-rib-attributes;

     organization
       "IETF IDR Working Group";
     contact
       "WG Web:   <http://datatracker.ietf.org/wg/idr>
        WG List:  <idr@ietf.org>

        Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                 Keyur Patel (keyur at arrcus.com),
                 Susan Hares (shares at ndzh.com),
                 Jeffrey Haas (jhaas at juniper dot net).";

     description
       "Defines a submodule for representing BGP Routing Information
        Base (RIB) contents.  The submodule supports 5 logical RIBs per
        address family:

        loc-rib: This is the main BGP routing table for the local
        routing instance, containing best-path selections for each
        prefix.  The key for the loc-rib will correspond either to a
        route in the adj-rib-in-post table, described below, or a route
        redistributed from another protocol.

        adj-rib-in-pre: This is a per-neighbor table containing the BGP
        routes received from the neighbor before any local input policy
        rules has been applied.  This can be considered the 'raw' routes
        from a given neighbor.

        adj-rib-in-post: This table is a per-neighbor table containing
        the routes received from the neighbor that are eligible for
        best-path selection after local input policy rules have been
        applied. The 'best-path' leaf is attached to the route selected
        by the BGP Decision Process (RFC 4271, Section 9.1).  Such
        routes may be present in the loc-rib table, described above,
        when local configuration determines that the BGP best-path will
        be used for that destination.

        adj-rib-in-post: This is a per-neighbor table containing the
        routes received from the neighbor that are eligible for
        best-path selection after local input policy rules have been
        applied.

        adj-rib-out-pre: This is a per-neighbor table containing routes
```

```
        eligible for sending (advertising) to the neighbor before output
        policy rules have been applied.

        adj-rib-out-post: This is a per-neighbor table containing routes
        eligible for sending (advertising) to the neighbor after output
        policy rules have been applied.

        Copyright (c) 2023 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 Revised 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 2023-07-05 {
       description
         "Initial Version";
       reference
         "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
     }

     grouping attr-set-attributes {
       description
         "A grouping for all BGP Path Attribute set parameters.";

       container attributes {
         description
           "A container for attribute set parameters.";

         leaf origin {
           type bt:bgp-origin-attr-type;
           description
             "BGP attribute defining the origin of the path
              information.";
           reference
```

```
            "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
             5.1.1.";
        }
        container as-path {
          description
            "Enclosing container for the list of AS path segments.

             In the Adj-RIB-In or Adj-RIB-Out, this list should show
             the received or sent AS_PATH, respectively.  For
             example, if the local router is not 4-byte capable, this
             value should consist of 2-octet ASNs or the AS_TRANS
             (AS 23456) values received or sent in BGP updates.

             In the Loc-RIB, this list should reflect the effective
             AS path for the route, e.g., a 4-octet value if the
             local router is 4-octet capable.";
          reference
            "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
             RFC 6793: BGP Support for Four-octet AS Number Space
             RFC 5065: Autonomous System Confederations for BGP.";
          list segment {
            config false;
            uses bgp-as-path-segment;
            description
              "List of AS PATH segments";
          }
        }
        leaf next-hop {
          type inet:ip-address;
          description
            "BGP next hop attribute defining the IP address of the
             router that should be used as the next hop to the
             destination.  Used when the BGP routes' nexthop for that
             AFI/SAFI can be represented as an IPv4/IPv6 address.";
          reference
            "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
             5.1.3.";
        }
        leaf link-local-next-hop {
          type inet:ipv6-address;
          description
            "When both a global and a link-local next-hop are sent
             when following RFC 2545 procedures, this leaf contains
             the link-local next-hop.";
          reference
            "RFC 2545: Use of BGP-4 Multiprotocol Extensions for IPv6
             Inter-Domain Routing.";
        }
```

```
        leaf med {
          type uint32;
          description
            "BGP multi-exit discriminator attribute used in the BGP
             route selection process.";
          reference
            "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
             5.1.4.";
        }
        leaf local-pref {
          type uint32;
          description
            "BGP local preference attribute sent to internal peers to
             indicate the degree of preference for externally learned
             routes.  The route with the highest local preference
             value is preferred.";
          reference
            "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
             5.1.5.";
        }
        container as4-path {
          description
            "This is the path encoded with 4-octet
             AS numbers in the optional transitive AS4_PATH attribute.
             This value is populated with the received or sent
             attribute in Adj-RIB-In or Adj-RIB-Out, respectively.
             It should not be populated in Loc-RIB since the Loc-RIB
             is expected to store the effective AS-Path in the
             as-path leaf regardless of being 4-octet or 2-octet.";
          reference
            "RFC 6793: BGP Support for Four-octet AS Number Space";
          list segment {
            config false;
            uses bgp-as-path-segment;
            description
              "List of AS PATH segments";
          }
        }
        container aggregator {
          config false;
          description
            "BGP attribute indicating the prefix has been
             aggregated by the specified AS and router.";
          reference
            "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
             5.1.7,
             RFC 6793: BGP Support for Four-octet AS Number Space.";
          leaf as {
```

```
            type inet:as-number;
            description
              "AS number of the autonomous system that performed the
               aggregation.";
            reference
              "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
               4.3, Path Attributes (g).";
          }
          leaf identifier {
            type yang:dotted-quad;
            description
              "BGP Identifier of the router that performed the
               aggregation.";
            reference
              "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
               4.3, Path Attributes (g).";
          }
        }
        container aggregator4 {
          config false;
          description
            "BGP attribute indicating the prefix has been
             aggregated by the specified AS and router.
             This value is populated with the received or sent
             attribute in Adj-RIB-In or Adj-RIB-Out, respectively.
             It should not be populated in Loc-RIB since the Loc-RIB
             is expected to store the effective AGGREGATOR in the
             aggregator/as leaf regardless of being 4-octet or
             2-octet.";
          reference
            "RFC 6793: BGP Support for Four-octet AS Number Space.";
          leaf as4 {
            type inet:as-number;
            description
              "AS number of the autonomous system that performed the
               aggregation (4-octet representation).  This value is
               populated if an upstream router is not 4-octet capable.
               Its semantics are similar to the AS4_PATH optional
               transitive attribute";
            reference
              "RFC 6793: BGP Support for Four-octet AS Number Space,
               Section 3,
               RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
               4.3, Path Attributes (g).";
          }
          leaf identifier {
            type yang:dotted-quad;
            description
```

```
              "BGP Identifier of the router that performed the
               aggregation.";
            reference
              "RFC 6793: BGP Support for Four-octet AS Number Space,
               Section 3,
               RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
               4.3, Path Attributes (g).";
          }
        }
        leaf atomic-aggregate {
          type boolean;
          description
            "BGP attribute indicating that the prefix is an atomic
             aggregate; i.e., the peer selected is a less specific
             route without selecting a more specific route that is
             subsumed by it.";
          reference
            "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section
             5.1.6.";
        }
        leaf originator-id {
          type yang:dotted-quad;
          description
            "BGP attribute that provides the id as an IPv4 address
             of the originator of the announcement.";
          reference
            "RFC 4456: BGP Route Reflection: An Alternative to Full
             Mesh Internal BGP (IBGP).";
        }
        leaf-list cluster-list {
          type yang:dotted-quad;
          description
            "Represents the reflection path that the route has
             passed.";
          reference
            "RFC 4456: BGP Route Reflection: An Alternative to Full
             Mesh Internal BGP (IBGP).";
        }
        leaf aigp-metric {
          type uint64;
          description
            "BGP path attribute representing the accumulated IGP
             metric for the path";
          reference
            "RFC 7311: The Accumulated IGP Metric Attribute for BGP.";
        }
      }
    }
```

```
     grouping attr-sets {
       description
         "A grouping for all sets of path attributes.";

       container attr-sets {
         description
           "Enclosing container for the list of path attribute sets";

         list attr-set {
           key "index";
           description
             "List of path attributes that may be in use by multiple
              routes in the table";
           leaf index {
             type uint64;
             description
               "System generated index for each attribute set.  The
                index is used to reference an attribute set from a
                specific path.  Multiple paths may reference the same
                attribute set.";
           }
           uses attr-set-attributes;
         }
       }
     }

     grouping rib {
       description
         "Grouping for rib.";

       container rib {
         config false;
         uses attr-sets;
         container communities {
           description
             "Enclosing container for the list of community attribute
              sets.";
           list community {
             key "index";
             config false;
             description
               "List of path attributes that may be in use by multiple
                routes in the table.";
             leaf index {
               type uint64;
               description
                 "System generated index for each attribute set.  The
                  index is used to reference an attribute set from a
```

```
                 specific path.  Multiple paths may reference the same
                 attribute set.";
             }
             uses bgp-community-attr-state;
           }
         }
         container ext-communities {
           description
             "Enclosing container for the list of extended community
              attribute sets.";
           list ext-community {
             key "index";
             config false;
             description
               "List of path attributes that may be in use by multiple
                routes in the table.";
             leaf index {
               type uint64;
               description
                 "System generated index for each attribute set.  The
                  index is used to reference an attribute set from a
                  specific path.  Multiple paths may reference the same
                  attribute set.";
             }
             uses ext-community-attributes;
           }
         }
         container ipv6-ext-communities {
           description
             "Enclosing container for the list of IPv6 extended
              community attribute sets.";
           list ipv6-ext-community {
             key "index";
             config false;
             description
               "List of path attributes that may be in use by multiple
                routes in the table.";
             leaf index {
               type uint64;
               description
                 "System generated index for each attribute set.  The
                  index is used to reference an attribute set from a
                  specific path.  Multiple paths may reference the same
                  attribute set.";
             }
             uses ipv6-ext-community-attributes;
           }
         }
```

```
        container large-communities {
          description
            "Enclosing container for the list of large community
             attribute sets.";
          list large-community {
            key "index";
            config false;
            description
              "List of path attributes that may be in use by multiple
               routes in the table.";
            leaf index {
              type uint64;
              description
                "System generated index for each attribute set. The
                 index is used to reference an attribute set from a
                 specific path.  Multiple paths may reference the same
                 attribute set.";
            }
            uses large-community-attributes;
          }
        }

        container afi-safis {
          config false;
          description
            "Enclosing container for address family list.";
          list afi-safi {
            key "name";
            description
              "List of afi-safi types.";
            leaf name {
              type identityref {
                base bt:afi-safi-type;
              }
              description
                "AFI,SAFI name.";
            }
            container ipv4-unicast {
              when "../name = 'bt:ipv4-unicast'" {
                description
                  "Include this container for IPv4 unicast RIB.";
              }
              description
                "Routing tables for IPv4 unicast -- active when the
                 afi-safi name is ipv4-unicast.";

              container loc-rib {
                config false;
```

```
                  description
                    "Container for the IPv4 Unicast BGP Loc-RIB data.";
                  container routes {
                    description
                      "Enclosing container for list of routes in the
                       routing table.";
                    list route {
                      key "prefix origin path-id";
                      description
                        "List of routes in the table, keyed by the route
                         prefix, the route origin, and path-id. The route
                         origin can be either the neighbor address from
                         which the route was learned, or the source
                         protocol that injected the route. The path-id
                         distinguishes routes for the same prefix
                         received from a neighbor (e.g., if add-paths is
                         enabled).";
                      leaf prefix {
                        type inet:ipv4-prefix;
                        description
                          "The IPv4 prefix corresponding to the route.";
                      }
                      uses bgp-loc-rib-common-keys;
                      uses bgp-loc-rib-common-attr-refs;
                      uses bgp-common-route-annotations-state;
                      uses bgp-unknown-attr-top;
                      uses rib-ext-route-annotations;
                    }
                  }
                }

                container neighbors {
                  config false;
                  description
                    "Enclosing container for neighbor list.";
                  list neighbor {
                    key "neighbor-address";
                    description
                      "List of neighbors (peers) of the local BGP
                       speaker.";
                    leaf neighbor-address {
                      type inet:ip-address;
                      description
                        "IP address of the BGP neighbor or peer.";
                    }
                    container adj-rib-in-pre {
                      description
                        "Per-neighbor table containing the BGP routes
```

```
                received from the neighbor before any local
                input policy rules or filters have been applied.
                This can be considered the 'raw' routes from
                the neighbor.";
            uses ipv4-adj-rib-common;
            uses clear-routes {
              description
                "Clears the adj-rib-in state for the containing
                 neighbor. Subsequently, implementations might
                 issue a 'route refresh' if 'route refresh' has
                 been negotiated, or reset the session. ";
            }
          }
          container adj-rib-in-post {
            description
              "Per-neighbor table containing the paths received
               from the neighbor that are eligible for
               best-path selection after local input policy
               rules have been applied.";
            uses ipv4-adj-rib-in-post;
            uses clear-routes {
              description
                "Clears the adj-rib-in state for the containing
                 neighbor. Subsequently, implementations might
                 issue a 'route refresh' if 'route refresh' has
                 been negotiated, or reset the session. ";
            }
          }
          container adj-rib-out-pre {
            description
              "Per-neighbor table containing paths eligible for
               sending (advertising) to the neighbor before
               output policy rules have been applied.";
            uses ipv4-adj-rib-common;
            uses clear-routes {
              description
                "Clears the adj-rib-out state for the
                 containing neighbor. Subsequently, neighbors
                 will announce BGP updates to resynchronize
                 these routes.";
            }
          }
          container adj-rib-out-post {
            description
              "Per-neighbor table containing paths eligible for
               sending (advertising) to the neighbor after
               output policy rules have been applied.";
            uses ipv4-adj-rib-common;
```

```
                    uses clear-routes {
                      description
                        "Clears the adj-rib-out state for the
                         containing neighbor. Subsequently, neighbors
                         will announce BGP updates to resynchronize
                         these routes.";
                    }
                  }
                }
              }
            }

            container ipv6-unicast {
              when "../name = 'bt:ipv6-unicast'" {
                description
                  "Include this container for IPv6 unicast RIB.";
              }
              description
                "Routing tables for IPv6 unicast -- active when the
                 afi-safi name is ipv6-unicast.";

              container loc-rib {
                config false;
                description
                  "Container for the IPv6 BGP Loc-RIB data.";
                container routes {
                  description
                    "Enclosing container for list of routes in the
                     routing table.";
                  list route {
                    key "prefix origin path-id";
                    description
                      "List of routes in the table, keyed by the route
                       prefix, the route origin, and path-id. The route
                       origin can be either the neighbor address from
                       which the route was learned, or the source
                       protocol that injected the route. The path-id
                       distinguishes routes for the same prefix
                       received from a neighbor (e.g., if add-paths is
                       enabled).";
                    leaf prefix {
                      type inet:ipv6-prefix;
                      description
                        "The IPv6 prefix corresponding to the route.";
                    }
                    uses bgp-loc-rib-common-keys;
                    uses bgp-loc-rib-common-attr-refs;
                    uses bgp-common-route-annotations-state;
```

```
                uses bgp-unknown-attr-top;
                uses rib-ext-route-annotations;
              }
            }
          }

          container neighbors {
            config false;
            description
              "Enclosing container for neighbor list.";
            list neighbor {
              key "neighbor-address";
              description
                "List of neighbors (peers) of the local BGP
                 speaker.";
              leaf neighbor-address {
                type inet:ip-address;
                description
                  "IP address of the BGP neighbor or peer.";
              }
              container adj-rib-in-pre {
                description
                  "Per-neighbor table containing the BGP routes
                   received from the neighbor before any local
                   input policy rules or filters have been applied.
                   This can be considered the 'raw' routes from
                   the neighbor.";
                uses ipv6-adj-rib-common;
                uses clear-routes {
                  description
                    "Clears the adj-rib-in state for the containing
                     neighbor. Subsequently, implementations might
                     issue a 'route refresh' if 'route refresh' has
                     been negotiated, or reset the session. ";
                }
              }
              container adj-rib-in-post {
                description
                  "Per-neighbor table containing the paths received
                   from the neighbor that are eligible for
                   best-path selection after local input policy
                   rules have been applied.";
                uses ipv6-adj-rib-in-post;
                uses clear-routes {
                  description
                    "Clears the adj-rib-in state for the containing
                     neighbor. Subsequently, implementations might
                     issue a 'route refresh' if 'route refresh' has
```

```
                        been negotiated, or reset the session. ";
                    }
                  }
                  container adj-rib-out-pre {
                    description
                      "Per-neighbor table containing paths eligible for
                       sending (advertising) to the neighbor before
                       output policy rules have been applied.";
                    uses ipv6-adj-rib-common;
                    uses clear-routes {
                      description
                        "Clears the adj-rib-out state for the
                         containing neighbor. Subsequently, neighbors
                         will announce BGP updates to resynchronize
                         these routes.";
                    }
                  }
                  container adj-rib-out-post {
                    description
                      "Per-neighbor table containing paths eligible for
                       sending (advertising) to the neighbor after
                       output policy rules have been applied.";
                    uses ipv6-adj-rib-common;
                    uses clear-routes {
                      description
                        "Clears the adj-rib-out state for the
                         containing neighbor. Subsequently, neighbors
                         will announce BGP updates to resynchronize
                         these routes.";
                    }
                  }
                }
              }
            }
          }
        }
        description
          "Top level container for BGP RIB.";
      }
    }
  }
  <CODE ENDS>
```

7.6.3. BGP RIB attributes submodule

```
<CODE BEGINS> file "ietf-bgp-rib-attributes@2023-07-05.yang"
submodule ietf-bgp-rib-attributes {
  yang-version 1.1;
  belongs-to ietf-bgp {
    prefix bgp;
  }

  // import some basic types

  import iana-bgp-types {
    prefix bt;
    reference
      "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
  }
  import iana-bgp-community-types {
    prefix bct;
    reference
      "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types.";
  }

  // meta

  organization
    "IETF IDR Working Group";
  contact
    "WG Web:   <http://datatracker.ietf.org/wg/idr>
     WG List:  <idr@ietf.org>

     Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
              Keyur Patel (keyur at arrcus.com),
              Susan Hares (shares at ndzh.com),
              Jeffrey Haas (jhaas at juniper.net).";

  description
    "This submodule contains common data definitions for BGP
     attributes for use in BGP RIB tables.

     Copyright (c) 2023 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 Revised 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 2023-07-05 {
       description
         "Initial version";
       reference
         "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
     }

     grouping bgp-as-path-segment {
       description
         "Data for representing BGP AS-PATH segment";

       leaf type {
         type identityref {
           base bt:as-path-segment-type;
         }
         description
           "The type of AS-PATH segment";
       }
       leaf-list member {
         type inet:as-number;
         description
           "List of the AS numbers in the AS-PATH segment";
       }
     }

     grouping bgp-unknown-attr-top {
       description
         "Unknown path attributes that are not expected to be shared
          across route entries, common to LOC-RIB and Adj-RIB";
       container unknown-attributes {
         description
           "Unknown path attributes that were received in the UPDATE
            message which contained the prefix.";
```

```
        list unknown-attribute {
          key "attr-type";
          description
            "This list contains received attributes that are
             unrecognized or unsupported by the local router. The list
             may be empty.";

          leaf optional {
            type boolean;
            description
              "Defines whether the attribute is optional (if
               set to true) or well-known (if set to false).
               Set in the high-order bit of the BGP attribute
               flags octet.";
            reference
              "RFC 4271: A Border Gateway Protocol 4 (BGP-4).";
          }

          leaf transitive {
            type boolean;
            description
              "Defines whether an optional attribute is transitive
               (if set to true) or non-transitive (if set to false).
               For well-known attributes, the transitive flag must be
               set to true. Set in the second high-order bit of the BGP
               attribute flags octet.";
            reference
              "RFC 4271: A Border Gateway Protocol 4 (BGP-4).";
          }

          leaf partial {
            type boolean;
            description
              "Defines whether the information contained in the
               optional transitive attribute is partial (if set to
               true) or complete (if set to false). For well-known
               attributes and for optional non-transitive attributes,
               the partial flag must be set to false. Set in the third
               high-order bit of the BGP attribute flags octet.";
            reference
              "RFC 4271: A Border Gateway Protocol 4 (BGP-4).";
          }

          leaf extended {
            type boolean;
            description
              "Defines whether the attribute length is one octet
               (if set to false) or two octets (if set to true). Set in
```

```
                the fourth high-order bit of the BGP attribute flags
                octet.";
            reference
              "RFC 4271: A Border Gateway Protocol 4 (BGP-4).";
          }

          leaf attr-type {
            type uint8;
            description
              "1-octet value encoding the attribute type code";
            reference
              "RFC 4271: A Border Gateway Protocol 4 (BGP-4).";
          }

          leaf attr-len {
            type uint16;
            description
              "One or two octet attribute length field indicating the
               length of the attribute data in octets.  If the Extended
               Length attribute flag is set, the length field is 2
               octets, otherwise it is 1 octet";
            reference
              "RFC 4271: A Border Gateway Protocol 4 (BGP-4).";
          }

          leaf attr-value {
            type binary {
              length "0..65535";
            }
            description
              "Raw attribute value, not including the attribute
               flags, type, or length.  The maximum length
               of the attribute value data is 2^16-1 per the max value
               of the attr-len field (2 octets).";
            reference
              "RFC 4271: A Border Gateway Protocol 4 (BGP-4).";
          }
        }
      }
    }

    grouping bgp-adj-rib-attr-state {
      description
        "Path attributes that are not expected to be shared across
         route entries, specific to Adj-RIB";
      leaf path-id {
        type uint32;
        description
```

```
          "When the BGP speaker supports advertisement of multiple
           paths for a prefix, the path identifier is used to
           uniquely identify a route based on the combination of the
           prefix and path id.  In the Adj-RIB-In, the path-id value is
           the value received in the update message.   In the Loc-RIB,
           if used, it should represent a locally generated path-id
           value for the corresponding route.  In Adj-RIB-Out, it
           should be the value sent to a neighbor when add-paths is
           used, i.e., the capability has been negotiated.";
        reference
          "RFC 7911: Advertisement of Multiple Paths in BGP.";
      }
    }

    grouping bgp-community-attr-state {
      description
        "Common definition of BGP community attributes";
      leaf-list community {
        type union {
          type bct:bgp-well-known-community-type;
          type bct:bgp-std-community-type;
        }
        description
          "List of standard or well-known BGP community
           attributes.";
      }
    }

    grouping ext-community-attributes {
      description
        "A grouping for all extended community parameters.";

      leaf-list ext-community {
        type bct:bgp-ext-community-type;
        description
          "List of BGP extended community attributes. The received
           extended community may be an explicitly modeled
           type or unknown, represented by an 8-octet value
           formatted according to RFC 4360.";
        reference
          "RFC 4360: BGP Extended Communities Attribute.";
      }

      leaf-list ext-community-raw {
        type string {
          // raw with 8 octets
          pattern 'raw:([0-9A-F][0-9A-F]:){7}[0-9A-F][0-9A-F]';
        }
```

```
        description
          "ext-community type in raw format only.";
      }
    }

    grouping ipv6-ext-community-attributes {
      description
        "A grouping for all IPv6 extended community parameters.";

      leaf-list ipv6-ext-community {
        type bct:bgp-ipv6-ext-community-type;
        description
          "List of BGP IPv6 extended community attributes. The received
           IPv6 extended community may be an explicitly modeled
           type or unknown, represented by an 20-octet value
           formatted according to RFC 5701.";
        reference
          "RFC 5701: IPv6 Address Specific BGP Extended Community
           Attribute, Section 3.";
      }

      leaf-list ipv6-ext-community-raw {
        type string {
          // raw with 20 octets
          pattern 'ipv6-raw:'
               + '([0-9A-Fa-f][0-9A-Fa-f]:){19}'
               + '[0-9A-Fa-f][0-9A-Fa-f]';
        }
        description
          "IPv6 ext-community type in raw format only.";
      }
    }

    grouping large-community-attributes {
      description
        "A grouping for all large community parameters.";

      leaf-list large-community {
        type bct:bgp-large-community-type;
        description
          "List of BGP large community attributes.";
        reference
          "RFC 8092: BGP Large Communities Attribute.";
      }
    }
  }
  <CODE ENDS>
```

7.6.4. BGP RIB tables submodule

```
<CODE BEGINS> file "ietf-bgp-rib-tables@2023-07-05.yang"
submodule ietf-bgp-rib-tables {
  yang-version 1.1;
  belongs-to ietf-bgp {
    prefix bgp;
  }

  // import some basic types

  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 {
    prefix rt;
    reference
      "RFC 8022: A YANG Data Model for Routing Management.";
  }
  import iana-bgp-types {
    prefix bt;
    reference
      "RFC XXXX: YANG module for Border Gateway Protocol (BGP-4).";
  }
  import iana-bgp-rib-types {
    prefix brt;
    reference
      "RFC XXXX: YANG module for Border Gateway Protocol (BGP-4).";
  }
  include ietf-bgp-rib-attributes;

  organization
    "IETF IDR Working Group";
  contact
    "WG Web:   <http://datatracker.ietf.org/wg/idr>
     WG List:  <idr@ietf.org>

     Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
              Keyur Patel (keyur at arrcus.com),
              Susan Hares (shares at ndzh.com,
              Jeffrey Haas (jhaas at juniper.net).";
```

```
     description
       "This submodule contains structural data definitions for
        BGP routing tables.

        Copyright (c) 2023 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 Revised 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 2023-07-05 {
       description
         "Initial Version";
       reference
         "RFC XXXX: YANG Model for Border Gateway Protocol (BGP-4).";
     }

     grouping bgp-common-route-annotations-state {
       description
         "Data definitions for flags and other information attached
          to routes in both LOC-RIB and Adj-RIB";
       leaf last-modified {
         type yang:timeticks;
         description
           "Timestamp when this path was last modified.

            The value is the timestamp in seconds relative to
            the Unix Epoch (Jan 1, 1970 00:00:00 UTC).";
       }
       leaf eligible-route {
         type boolean;
         description
           "Indicates that the route is eligible for selection for the
            best route in the Loc-Rib in BGP's Decision Process.";
```

```
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
           Section 9.1.";
      }
      leaf ineligible-reason {
        type identityref {
          base brt:ineligible-route-reason;
        }
        description
          "If the route is ineligible for selection for the best route
           in the Loc-Rib in BGP's Decision process, this indicates the
           reason.";
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
           Section 9.1.";
      }
    }

    grouping bgp-adj-rib-in-post-route-annotations-state {
      description
        "Data definitions for information attached to routes in the
         Adj-RIB-In post-policy table";
      leaf best-path {
        type boolean;
        default "false";
        description
          "Current path was selected as the best path. Best path
           should indicate that the route is present in BGP Loc-RIB.";
      }
    }

    grouping rib-ext-route-annotations {
      description
        "Extended annotations for routes in the routing tables";
      leaf reject-reason {
        type union {
          type identityref {
            base brt:bgp-not-selected-bestpath;
          }
          type identityref {
            base brt:bgp-not-selected-policy;
          }
        }
        description
          "Indicates the reason the route is not used, either due to
           policy filtering or bestpath selection";
      }
    }
```

```
     grouping bgp-adj-rib-common-attr-refs {
       description
         "Definitions of common references to attribute sets for
          multiple AFI-SAFIs for Adj-RIB tables.";
       leaf attr-index {
         type leafref {
           path "../../../../../../../../../attr-sets/"
              + "attr-set/index";
         }
         description
           "Reference to the common attribute group for the
            route.";
       }
       leaf community-index {
         type leafref {
           path "../../../../../../../../../communities/community/"
              + "index";
         }
         description
           "Reference to the community attribute for the route.";
       }
       leaf ext-community-index {
         type leafref {
           path "../../../../../../../../../ext-communities/"
              + "ext-community/index";
         }
         description
           "Reference to the extended community attribute for the
            route.";
       }
       leaf large-community-index {
         type leafref {
           path "../../../../../../../../../large-communities/"
              + "large-community/index";
         }
         description
           "Reference to the large community attribute for the
            route.";
       }
     }

     grouping bgp-loc-rib-common-attr-refs {
       description
         "Definitions of common references to attribute sets for
          multiple AFI-SAFIs for Loc-RIB tables.";
       leaf attr-index {
         type leafref {
           path "../../../../../../../attr-sets/attr-set/"
```

```
              + "index";
          }
          description
            "Reference to the common attribute group for the
             route.";
        }
        leaf community-index {
          type leafref {
            path "../../../../../../../communities/community/"
               + "index";
          }
          description
            "Reference to the community attribute for the route.";
        }
        leaf ext-community-index {
          type leafref {
            path "../../../../../../../ext-communities/"
               + "ext-community/index";
          }
          description
            "Reference to the extended community attribute for the
             route.";
        }
        leaf large-community-index {
          type leafref {
            path "../../../../../../../large-communities/"
               + "large-community/index";
          }
          description
            "Reference to the large community attribute for the
             route.";
        }
      }

      grouping bgp-loc-rib-common-keys {
        description
          "Common references used in keys for IPv4 and IPv6
           Loc-RIB entries.";
        leaf origin {
          type union {
            type inet:ip-address;
            type identityref {
              base rt:routing-protocol;
            }
          }
          description
            "Indicates the origin of the route.  If the route is learned
             from a neighbor, this value is the neighbor address.  If
```

```
            the route was injected or redistributed from another
            protocol, the origin indicates the source protocol for the
            route.";
        }
        leaf path-id {
          type uint32;
          description
            "If the route is learned from a neighbor, the path-id
             corresponds to the path-id for the route in the
             corresponding adj-rib-in-post table.  If the route is
             injected from another protocol, or the neighbor does not
             support BGP add-paths, the path-id should be set
             to zero, also the default value.

             However, YANG does not allow default values to be set
             for parameters that form the key, so a default value
             cannot be set here.";
        }
      }

      grouping clear-routes {
        description
          "Action to clear BGP routes.";
        container clear-routes {
          if-feature "bt:clear-routes";
          action clear {
            input {
              leaf clear-at {
                type yang:date-and-time;
                description
                  "The time, in the future when the clear operation will
                   be initiated.";
              }
            }
            output {
              leaf clear-finished-at {
                type yang:date-and-time;
                description
                  "The time when the clear operation finished.";
              }
            }
          }
          description
            "Action commands to clear routes governed by a if-feature.";
        }
      }

      grouping ipv4-adj-rib-common {
```

```
      description
        "Common structural grouping for each IPv4 Adj-RIB table.";
      container routes {
        config false;
        description
          "Enclosing container for list of routes in the routing
           table.";
        list route {
          key "prefix path-id";
          description
            "List of routes in the table, keyed by a combination of
             the route prefix and path-id to distinguish multiple
             routes received from a neighbor for the same prefix;
             e.g., when BGP add-paths is enabled.";
          leaf prefix {
            type inet:ipv4-prefix;
            description
              "Prefix for the route.";
          }
          uses bgp-adj-rib-attr-state;
          uses bgp-adj-rib-common-attr-refs;
          uses bgp-common-route-annotations-state;
          uses bgp-unknown-attr-top;
          uses rib-ext-route-annotations;
        }
      }
    }

    grouping ipv4-adj-rib-in-post {
      description
        "Common structural grouping for the IPv4 Adj-RIB-In
         post-policy table.";
      container routes {
        config false;
        description
          "Enclosing container for list of routes in the routing
           table.";
        list route {
          key "prefix path-id";
          description
            "List of routes in the table, keyed by a combination of
             the route prefix and path-id to distinguish multiple
             routes received from a neighbor for the same prefix;
             e.g., when BGP add-paths is enabled.";
          leaf prefix {
            type inet:ipv4-prefix;
            description
              "Prefix for the route.";
```

```
          }
          uses bgp-adj-rib-attr-state;
          uses bgp-adj-rib-common-attr-refs;
          uses bgp-common-route-annotations-state;
          uses bgp-adj-rib-in-post-route-annotations-state;
          uses bgp-unknown-attr-top;
          uses rib-ext-route-annotations;
        }
      }
    }

    grouping ipv6-adj-rib-common {
      description
        "Common structural grouping for each IPv6 Adj-RIB table.";
      container routes {
        config false;
        description
          "Enclosing container for list of routes in the routing
           table.";
        list route {
          key "prefix path-id";
          description
            "List of routes in the table.";
          leaf prefix {
            type inet:ipv6-prefix;
            description
              "Prefix for the route.";
          }
          uses bgp-adj-rib-attr-state;
          uses bgp-adj-rib-common-attr-refs;
          uses bgp-common-route-annotations-state;
          uses bgp-unknown-attr-top;
          uses rib-ext-route-annotations;
        }
      }
    }

    grouping ipv6-adj-rib-in-post {
      description
        "Common structural grouping for the IPv6 Adj-RIB-In
         post-policy table.";
      container routes {
        config false;
        description
          "Enclosing container for list of routes in the routing
           table.";
        list route {
          key "prefix path-id";
```

```
          description
            "List of routes in the table.";
          leaf prefix {
            type inet:ipv6-prefix;
            description
              "Prefix for the route.";
          }
          uses bgp-adj-rib-attr-state;
          uses bgp-adj-rib-common-attr-refs;
          uses bgp-common-route-annotations-state;
          uses bgp-adj-rib-in-post-route-annotations-state;
          uses bgp-unknown-attr-top;
          uses rib-ext-route-annotations;
        }
      }
    }
  }
  <CODE ENDS>
```

8. Contributors

Previous versions of this document saw contributions from Anees Shaikh, Rob Shakir, Kevin D'Souza, Alexander Clemm, Aleksandr Zhadkin, and Xufeng Liu.

9. Acknowledgements

The authors are grateful for valuable contributions to this document and the associated models from: Ebben Aires, Pavan Beeram, Chris Chase, Ed Crabbe, Luyuan Fang, Bill Fenner, Akshay Gattani, Josh George, Vijay Gill, Matt John, Dhanendra Jain, Acee Lindem, Ina Minei, Carl Moberg, Ashok Narayanan, Einar Nilsen-Nygaard, Adam Simpson, Puneet Sood, Jason Sterne, Jeff Tantsura, Jim Uttaro, and Gunter Vandevelde.

Credit is also due to authors of the OpenConfig, whose model was relied upon to come up with this model.

Special thanks to Robert Wilton who helped convert the YANG models to a NMDA compatible model, to Yingzhen Qu who contributed to the policy part of the module, and to Derek Yeung in helping out with Extended Communities support in the model.

10. References

10.1. Normative references

[RFC1997] Chandra, R., Traina, P., and T. Li, "BGP Communities Attribute", RFC 1997, DOI 10.17487/RFC1997, August 1996, <https://www.rfc-editor.org/info/rfc1997>.

[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>.

[RFC2439] Villamizar, C., Chandra, R., and R. Govindan, "BGP Route Flap Damping", RFC 2439, DOI 10.17487/RFC2439, November 1998, <https://www.rfc-editor.org/info/rfc2439>.

[RFC2918] Chen, E., "Route Refresh Capability for BGP-4", RFC 2918, DOI 10.17487/RFC2918, September 2000, <https://www.rfc-editor.org/info/rfc2918>.

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

[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>.

[RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, DOI 10.17487/RFC4360, February 2006, <https://www.rfc-editor.org/info/rfc4360>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

[RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, DOI 10.17487/RFC4456, April 2006, <https://www.rfc-editor.org/info/rfc4456>.

[RFC4659] De Clercq, J., Ooms, D., Carugi, M., and F. Le Faucheur, "BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN", RFC 4659, DOI 10.17487/RFC4659, September 2006, <https://www.rfc-editor.org/info/rfc4659>.

[RFC4724] Sangli, S., Chen, E., Fernando, R., Scudder, J., and Y. Rekhter, "Graceful Restart Mechanism for BGP", RFC 4724, DOI 10.17487/RFC4724, January 2007, <https://www.rfc-editor.org/info/rfc4724>.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <https://www.rfc-editor.org/info/rfc4760>.

[RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <https://www.rfc-editor.org/info/rfc4761>.

[RFC5065] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", RFC 5065, DOI 10.17487/RFC5065, August 2007, <https://www.rfc-editor.org/info/rfc5065>.

[RFC5701] Rekhter, Y., "IPv6 Address Specific BGP Extended Community Attribute", RFC 5701, DOI 10.17487/RFC5701, November 2009, <https://www.rfc-editor.org/info/rfc5701>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, DOI 10.17487/RFC5881, June 2010, <https://www.rfc-editor.org/info/rfc5881>.

[RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", RFC 5883, DOI 10.17487/RFC5883, June 2010, <https://www.rfc-editor.org/info/rfc5883>.

[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>.

[RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", RFC 6514, DOI 10.17487/RFC6514, February 2012, <https://www.rfc-editor.org/info/rfc6514>.

[RFC6793] Vohra, Q. and E. Chen, "BGP Support for Four-Octet Autonomous System (AS) Number Space", RFC 6793, DOI 10.17487/RFC6793, December 2012, <https://www.rfc-editor.org/info/rfc6793>.

[RFC6811] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, "BGP Prefix Origin Validation", RFC 6811, DOI 10.17487/RFC6811, January 2013, <https://www.rfc-editor.org/info/rfc6811>.

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

[RFC7607] Kumari, W., Bush, R., Schiller, H., and K. Patel, "Codification of AS 0 Processing", RFC 7607, DOI 10.17487/RFC7607, August 2015, <https://www.rfc-editor.org/info/rfc7607>.

[RFC7911] Walton, D., Retana, A., Chen, E., and J. Scudder, "Advertisement of Multiple Paths in BGP", RFC 7911, DOI 10.17487/RFC7911, July 2016, <https://www.rfc-editor.org/info/rfc7911>.

[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>.

[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>.

[RFC8092] Heitz, J., Ed., Snijders, J., Ed., Patel, K., Bagdonas, I., and N. Hilliard, "BGP Large Communities Attribute", RFC 8092, DOI 10.17487/RFC8092, February 2017, <https://www.rfc-editor.org/info/rfc8092>.

[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>.

[RFC8277] Rosen, E., "Using BGP to Bind MPLS Labels to Address Prefixes", RFC 8277, DOI 10.17487/RFC8277, October 2017, <https://www.rfc-editor.org/info/rfc8277>.

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

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

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

[RFC8528] Bjorklund, M. and L. Lhotka, "YANG Schema Mount", RFC 8528, DOI 10.17487/RFC8528, March 2019, <https://www.rfc-editor.org/info/rfc8528>.

[RFC8529] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Data Model for Network Instances", RFC 8529, DOI 10.17487/RFC8529, March 2019, <https://www.rfc-editor.org/info/rfc8529>.

[RFC9067] Qu, Y., Tantsura, J., Lindem, A., and X. Liu, "A YANG Data Model for Routing Policy", RFC 9067, DOI 10.17487/RFC9067, October 2021, <https://www.rfc-editor.org/info/rfc9067>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

[RFC9314] Jethanandani, M., Ed., Rahman, R., Ed., Zheng, L., Ed., Pallagatti, S., and G. Mirsky, "YANG Data Model for Bidirectional Forwarding Detection (BFD)", RFC 9314, DOI 10.17487/RFC9314, September 2022, <https://www.rfc-editor.org/info/rfc9314>.

[I-D.ietf-tcpm-yang-tcp]
Scharf, M., Jethanandani, M., and V. Murgai, "A YANG Model for Transmission Control Protocol (TCP) Configuration and State", Work in Progress, Internet-Draft, draft-ietf-tcpm-yang-tcp-09, 11 September 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-tcpm-yang-tcp-09>.

10.2. Informative references

[RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC 2385, DOI 10.17487/RFC2385, August 1998, <https://www.rfc-editor.org/info/rfc2385>.

[RFC2622] Alaettinoglu, C., Villamizar, C., Gerich, E., Kessens, D., Meyer, D., Bates, T., Karrenberg, D., and M. Terpstra, "Routing Policy Specification Language (RPSL)", RFC 2622, DOI 10.17487/RFC2622, June 1999, <https://www.rfc-editor.org/info/rfc2622>.

[RFC3765] Huston, G., "NOPEER Community for Border Gateway Protocol (BGP) Route Scope Control", RFC 3765, DOI 10.17487/RFC3765, April 2004, <https://www.rfc-editor.org/info/rfc3765>.

[RFC4451] McPherson, D. and V. Gill, "BGP MULTI_EXIT_DISC (MED) Considerations", RFC 4451, DOI 10.17487/RFC4451, March 2006, <https://www.rfc-editor.org/info/rfc4451>.

[RFC5082] Gill, V., Heasley, J., Meyer, D., Savola, P., Ed., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, DOI 10.17487/RFC5082, October 2007, <https://www.rfc-editor.org/info/rfc5082>.

[RFC5398] Huston, G., "Autonomous System (AS) Number Reservation for Documentation Use", RFC 5398, DOI 10.17487/RFC5398, December 2008, <https://www.rfc-editor.org/info/rfc5398>.

[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>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

[RFC7454] Durand, J., Pepelnjak, I., and G. Doering, "BGP Operations and Security", BCP 194, RFC 7454, DOI 10.17487/RFC7454, February 2015, <https://www.rfc-editor.org/info/rfc7454>.

[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>.

[I-D.ietf-idr-deprecate-as-set-confed-set] Kumari, W. A., Sriram, K., Hannachi, L., and J. Haas, "Deprecation of AS_SET and AS_CONFED_SET in BGP", Work in Progress, Internet-Draft, draft-ietf-idr-deprecate-as-set-confed-set-10, 16 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-deprecate-as-set-confed-set-10>.

Appendix A. Examples

This section tries to show some examples in how the model can be used.

A.1. Creating BGP Instance

This example shows how to enable BGP for a IPv4 unicast address family.

[note: '\' line wrapping for formatting only]

```
<?xml version="1.0" encoding="UTF-8"?>
<routing
    xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
  <control-plane-protocols>
    <control-plane-protocol>
      <type
          xmlns:bgp="urn:ietf:params:xml:ns:yang:ietf-bgp">bgp:bgp</\
type>
      <name>BGP</name>
      <bgp
          xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
        <global>
          <as>64496</as>
          <afi-safis>
            <afi-safi>
              <name
                  xmlns:bt=
                  "urn:ietf:params:xml:ns:yang:iana-bgp-types">bt:ip\
v4-unicast</name>
            </afi-safi>
          </afi-safis>
        </global>
      </bgp>
    </control-plane-protocol>
  </control-plane-protocols>
</routing>
```

A.2. Neighbor Address Family Configuration

This example shows how to configure a BGP neighbor, where the remote address is 192.0.2.1, the remote AS number is 64497, and the address family of the neighbor is IPv4 unicast. The neighbor is configured for route flap prevention and it set up for standard and large communities. In addition, BFD is configured at a neighbor level with a local multiplier of 2, a desired minimum transmit interval, and a required minimum receive interval of 3.3 ms.

[note: '\' line wrapping for formatting only]

```
<!--
    This example shows a neighbor configuration with damping.
-->

<?xml version="1.0" encoding="UTF-8"?>
<routing
    xmlns="urn:ietf:params:xml:ns:yang:ietf-routing"
```

```
      xmlns:bt="urn:ietf:params:xml:ns:yang:iana-bgp-types"
      xmlns:bct="urn:ietf:params:xml:ns:yang:iana-bgp-community-types"\
   >
     <control-plane-protocols>
       <control-plane-protocol>
         <type
             xmlns:bgp="urn:ietf:params:xml:ns:yang:ietf-bgp">bgp:bgp</\
   type>
         <name>name:BGP</name>
         <bgp
             xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
           <global>
             <as>64496</as>
             <afi-safis>
               <afi-safi>
                 <name>bt:ipv4-unicast</name>
               </afi-safi>
             </afi-safis>
           </global>
           <neighbors>
             <neighbor>
               <remote-address>192.0.2.1</remote-address>
               <peer-as>64497</peer-as>
               <route-flap-damping>
                 <enable>true</enable>
                 <suppress-above>4.0</suppress-above>
                 <reuse-above>3.0</reuse-above>
                 <max-flap>15.0</max-flap>
                 <reach-decay>100</reach-decay>
                 <unreach-decay>500</unreach-decay>
                 <keep-history>1000</keep-history>
               </route-flap-damping>
               <send-community>bct:standard</send-community>
               <send-community>bct:large</send-community>
               <description>"Peer Router B"</description>
               <afi-safis>
                 <afi-safi>
                   <name>bt:ipv4-unicast</name>
                 </afi-safi>
               </afi-safis>
               <transport>
                 <tcp-mss>1400</tcp-mss>
                 <mtu-discovery>true</mtu-discovery>
                 <bfd>
                   <enabled>true</enabled>
                   <local-multiplier>2</local-multiplier>
                   <desired-min-tx-interval>3300</desired-min-tx-interv\
   al>
```

```
                  <required-min-rx-interval>3300</required-min-rx-inte\
rval>
              </bfd>
            </transport>
          </neighbor>
        </neighbors>
      </bgp>
    </control-plane-protocol>
  </control-plane-protocols>
</routing>
```

A.3. IPv6 Neighbor Configuration

This example shows how to configure a BGP peer, where the remote peer has a IPv6 address, uses TCP-AO to secure the session with the peer, and uses non-default timers for hold-time and keepalive.

[note: '\' line wrapping for formatting only]

```
<?xml version="1.0" encoding="UTF-8"?>
<key-chains
    xmlns="urn:ietf:params:xml:ns:yang:ietf-key-chain">
  <key-chain>
    <name>bgp-key-chain</name>
    <description>"An example of TCP-AO configuration for BGP"</descr\
iption>
    <key>
      <key-id>55</key-id>
      <lifetime>
        <send-lifetime>
          <start-date-time>2017-01-01T00:00:00Z</start-date-time>
          <end-date-time>2017-02-01T00:00:00Z</end-date-time>
        </send-lifetime>
        <accept-lifetime>
          <start-date-time>2016-12-31T23:59:55Z</start-date-time>
          <end-date-time>2017-02-01T00:00:05Z</end-date-time>
        </accept-lifetime>
      </lifetime>
      <crypto-algorithm>aes-cmac-prf-128</crypto-algorithm>
      <key-string>
        <keystring>testvector</keystring>
      </key-string>
      <authentication
          xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp">
        <keychain>bgp-key-chain</keychain>
        <ao>
          <send-id>65</send-id>
          <recv-id>87</recv-id>
```

```
          </ao>
        </authentication>
      </key>
      <key>
        <key-id>56</key-id>
        <lifetime>
          <send-lifetime>
            <start-date-time>2017-01-01T00:00:00Z</start-date-time>
            <end-date-time>2017-02-01T00:00:00Z</end-date-time>
          </send-lifetime>
          <accept-lifetime>
            <start-date-time>2016-12-31T23:59:55Z</start-date-time>
            <end-date-time>2017-02-01T00:00:05Z</end-date-time>
          </accept-lifetime>
        </lifetime>
        <crypto-algorithm>aes-cmac-prf-128</crypto-algorithm>
        <key-string>
          <keystring>testvector</keystring>
        </key-string>
        <authentication
            xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp">
          <keychain>bgp-key-chain</keychain>
          <ao>
            <send-id>65</send-id>
            <recv-id>87</recv-id>
          </ao>
        </authentication>
      </key>
    </key-chain>
  </key-chains>

  <routing
      xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type
            xmlns:bgp="urn:ietf:params:xml:ns:yang:ietf-bgp">bgp:bgp</\
  type>
        <name>name:BGP</name>
        <bgp
            xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
          <global>
            <as>64496</as>
            <afi-safis>
              <afi-safi>
                <name
                    xmlns:bt=
                    "urn:ietf:params:xml:ns:yang:iana-bgp-types">bt:ip\
```

```
   v6-unicast</name>
               </afi-safi>
             </afi-safis>
           </global>
           <neighbors>
             <neighbor>
               <remote-address>2001:db8::</remote-address>
               <enabled>true</enabled>
               <transport>
                 <secure-session>
                   <enabled>true</enabled>
                   <options>
                     <ao-keychain>bgp-key-chain</ao-keychain>
                   </options>
                 </secure-session>
               </transport>
               <peer-as>64497</peer-as>
               <description>"Peer Router B"</description>
               <timers>
                 <hold-time>120</hold-time>
                 <keepalive>70</keepalive>
               </timers>
               <afi-safis>
                 <afi-safi>
                   <name
                       xmlns:bt=
                       "urn:ietf:params:xml:ns:yang:iana-bgp-types">bt:\
   ipv6-unicast</name>
                 </afi-safi>
               </afi-safis>
             </neighbor>
           </neighbors>
         </bgp>
       </control-plane-protocol>
     </control-plane-protocols>
   </routing>
```

A.4. VRF Configuration

This example shows how BGP can be configured for two VRFs, red and blue. In this case, the two network instances share a common AS, and distinguish between the instances using the router id.

[note: '\' line wrapping for formatting only]

```
<?xml version="1.0" encoding="UTF-8"?>
<network-instances
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
  <network-instance>
    <name>vrf-red</name>
    <vrf-root>
      <routing
          xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
        <router-id>192.0.2.1</router-id>
        <control-plane-protocols>
          <control-plane-protocol>
            <type
                xmlns:bgp=
                "urn:ietf:params:xml:ns:yang:ietf-bgp">bgp:bgp</type\
>
            <name>BGP</name>
            <bgp
                xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
              <global>
                <as>64496</as>
                <afi-safis>
                  <afi-safi>
                    <name
                        xmlns:bt=
                        "urn:ietf:params:xml:ns:yang:iana-bgp-types"\
>bt:ipv4-unicast</name>
                  </afi-safi>
                </afi-safis>
              </global>
            </bgp>
          </control-plane-protocol>
        </control-plane-protocols>
      </routing>
    </vrf-root>
  </network-instance>
  <network-instance>
    <name>vrf-blue</name>
    <vrf-root>
      <routing
          xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
        <router-id>192.0.2.2</router-id>
        <control-plane-protocols>
          <control-plane-protocol>
            <type
                xmlns:bgp=
                "urn:ietf:params:xml:ns:yang:ietf-bgp">bgp:bgp</type\
```

```
>
            <name>BGP</name>
            <bgp
                xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
              <global>
                <as>64496</as>
                <afi-safis>
                  <afi-safi>
                    <name
                        xmlns:bt=
                        "urn:ietf:params:xml:ns:yang:iana-bgp-types"\
>bt:ipv4-unicast</name>
                  </afi-safi>
                </afi-safis>
              </global>
            </bgp>
          </control-plane-protocol>
        </control-plane-protocols>
      </routing>
    </vrf-root>
  </network-instance>
</network-instances>
```

A.5. BGP Policy - Match Prefix and Set Community

Routing policy using community value involves configuring rules to match community values in the inbound or outbound direction. In this example, which is heavily borrowed from the example on the Cisco community page, we look at "match community exact" match, which happens only when BGP updates have the same community values as specified in the community list.

The topology in this example consists of three routers, R1, R2, and R3, configured with AS value of 1, 2 and 3 respectively. R1 advertises 5 prefixes to R2 and R3, as shown below.

* 1.1.1.1/32 and 2.2.2.2/32 with community 11:11

* 3.3.3.3/32 and 4.4.4.4/32 with community 11:11 and 22:22

* 5.5.5.5/32 with community 33.33

Route Policy TO_R2 defines the policy that R1 uses in route updates towards R2. It consists of three statements, statement 10 that has a exact match rule for the prefix list L0andL1, and a set-community action of add for 11:11. The second statement, statement 20, consists of an exact match rule for prefix list L2andL3, with a set community action of remove for 11:11 22:22. The final statement, statement 30, consists of an exact match rule for prefix list L4, with a set community action of replace for 33:33.

[note: '\' line wrapping for formatting only]

```
<?xml version="1.0" encoding="UTF-8"?>
<routing-policy
    xmlns="urn:ietf:params:xml:ns:yang:ietf-routing-policy">
  <defined-sets>
    <prefix-sets>
      <prefix-set>
        <name>L0andL1</name>
        <mode>ipv4</mode>
        <prefixes>
          <prefix-list>
            <ip-prefix>1.1.1.1/32</ip-prefix>
            <mask-length-lower>32</mask-length-lower>
            <mask-length-upper>32</mask-length-upper>
          </prefix-list>
          <prefix-list>
            <ip-prefix>2.2.2.2/32</ip-prefix>
            <mask-length-lower>32</mask-length-lower>
            <mask-length-upper>32</mask-length-upper>
          </prefix-list>
        </prefixes>
      </prefix-set>
      <prefix-set>
        <name>L2andL3</name>
        <mode>ipv4</mode>
        <prefixes>
          <prefix-list>
            <ip-prefix>3.3.3.3/32</ip-prefix>
            <mask-length-lower>32</mask-length-lower>
            <mask-length-upper>32</mask-length-upper>
          </prefix-list>
          <prefix-list>
            <ip-prefix>4.4.4.4/32</ip-prefix>
            <mask-length-lower>32</mask-length-lower>
            <mask-length-upper>32</mask-length-upper>
          </prefix-list>
        </prefixes>
      </prefix-set>
```

```
        <prefix-set>
          <name>L4</name>
          <mode>ipv4</mode>
          <prefixes>
            <prefix-list>
              <ip-prefix>5.5.5.5/32</ip-prefix>
              <mask-length-lower>32</mask-length-lower>
              <mask-length-upper>32</mask-length-upper>
            </prefix-list>
          </prefixes>
        </prefix-set>
      </prefix-sets>
    </defined-sets>
    <policy-definitions>
      <policy-definition>
        <name>TO_R2</name>
        <statements>
          <statement>
            <name>10</name>
            <conditions>
              <match-prefix-set>
                <prefix-set>L0andL1</prefix-set>
              </match-prefix-set>
            </conditions>
            <actions>
              <bgp-actions
                  xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp-policy">\

                <set-community>
                  <options>add</options>
                  <communities>11:11</communities>
                </set-community>
              </bgp-actions>
            </actions>
          </statement>
          <statement>
            <name>20</name>
            <conditions>
              <match-prefix-set>
                <prefix-set>L2andL3</prefix-set>
              </match-prefix-set>
            </conditions>
            <actions>
              <bgp-actions
                  xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp-policy">\

                <set-community>
                  <options>remove</options>
```

```
                  <communities>11:11</communities>
                  <communities>22:22</communities>
                </set-community>
              </bgp-actions>
            </actions>
          </statement>
          <statement>
            <name>30</name>
            <conditions>
              <match-prefix-set>
                <prefix-set>L4</prefix-set>
              </match-prefix-set>
            </conditions>
            <actions>
              <bgp-actions
                  xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp-policy">\

                <set-community>
                  <options>replace</options>
                  <communities>33:33</communities>
                </set-community>
              </bgp-actions>
            </actions>
          </statement>
        </statements>
      </policy-definition>
    </policy-definitions>
  </routing-policy>

  <routing
      xmlns="urn:ietf:params:xml:ns:yang:ietf-routing"
      xmlns:bt="urn:ietf:params:xml:ns:yang:iana-bgp-types"
      xmlns:bct="urn:ietf:params:xml:ns:yang:iana-bgp-community-types"\
  >
    <control-plane-protocols>
      <control-plane-protocol>
        <type
            xmlns:bgp="urn:ietf:params:xml:ns:yang:ietf-bgp">bgp:bgp</\
  type>
        <name>BGP</name>
        <bgp
            xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
          <global>
            <as>1</as>
            <afi-safis>
              <afi-safi>
                <name>bt:ipv4-unicast</name>
              </afi-safi>
```

```
            </afi-safis>
          </global>
          <neighbors>
            <neighbor>
              <remote-address>10.1.1.2</remote-address>
              <peer-as>2</peer-as>
              <afi-safis>
                <afi-safi>
                  <name>bt:ipv4-unicast</name>
                </afi-safi>
              </afi-safis>
              <send-community>bct:standard</send-community>
              <apply-policy>
                <export-policy>TO_R2</export-policy>
                <default-export-policy>accept-route</default-export-po\
   licy>
              </apply-policy>
            </neighbor>
          </neighbors>
        </bgp>
      </control-plane-protocol>
    </control-plane-protocols>
   </routing>
```

A.6. BGP Policy - Match Next-hop and Set Community

In this example, a "next-hop-set" is configured using option "invert" from "matct-set-options" to match next-hop not equal to "192.0.2.2" and then set community to 55:55.

[note: '\' line wrapping for formatting only]

```
<?xml version='1.0' encoding='UTF-8'?>
  <routing-policy xmlns="urn:ietf:params:xml:ns:yang:ietf-routing-po\
licy">
    <defined-sets>
      <bgp-defined-sets xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp-\
policy">
        <next-hop-sets>
          <next-hop-set>
            <name>nexthop_not2</name>
            <next-hop>192.0.2.2</next-hop>
          </next-hop-set>
        </next-hop-sets>
      </bgp-defined-sets>
    </defined-sets>
    <policy-definitions>
      <policy-definition>
        <name>nexthop_not2_community</name>
        <statements>
          <statement>
            <name>100</name>
            <conditions>
              <bgp-conditions xmlns="urn:ietf:params:xml:ns:yang:iet\
f-bgp-policy">
                <match-next-hop-set>
                  <next-hop-set>nexthop_not2</next-hop-set>
                  <match-set-options>invert</match-set-options>
                </match-next-hop-set>
              </bgp-conditions>
            </conditions>
            <actions>
              <bgp-actions xmlns="urn:ietf:params:xml:ns:yang:ietf-b\
gp-policy">
                <set-community>
                  <options>add</options>
                  <communities>55:55</communities>                  \

                </set-community>
              </bgp-actions>
            </actions>
          </statement>
        </statements>
      </policy-definition>
    </policy-definitions>
  </routing-policy>
```

Appendix B. How to add a new AFI and Augment a Module

This section explains how a new AFI can be defined in a new module and how that module can then be augmented. Assume that the new AFI being defined is called 'foo' which extends the base identity of 'afi-safi-type', and the augmentation is to add a new container for 'foo' under two different XPaths. The example shows how the base identity can be extended to add this new AFI, and then use the augmented containers be used to add 'foo' specific information.

```
module example-newafi-bgp {
  yang-version 1.1;
  namespace "http://example.com/ns/example-newafi-bgp";
  prefix example-newafi-bgp;

  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-bgp {
    prefix "bgp";
    reference
      "RFC XXXX: YANG module for Border Gateway Protocol (BGP).";
  }

  import iana-bgp-types {
    prefix "bt";
  }

  organization
    "Newafi model group.";

  contact
    "abc@newafi.com";
  description
    "This YANG module defines and uses new AFI.";

  revision 2023-07-05 {
    description
```

```
        "Creating new AFI and using in this model";

      reference
        "RFC XXXX: YANG module for Border Gateway Protocol (BGP).";
    }

    identity foo {
      base bt:afi-safi-type;
      description
        "New AFI type foo.";
    }

    augment "/rt:routing/rt:control-plane-protocols/" +
            "rt:control-plane-protocol/bgp:bgp/bgp:global/" +
            "bgp:afi-safis/bgp:afi-safi" {
      when "derived-from-or-self(bgp:name, 'foo')" {
        description
          "This augmentation is valid for a AFI/SAFI instance
           of 'foo'";
      }
      container foo {
        description
          "Container to add 'foo' specific AFI/SAFI information.
           First add the common stuff.";
        uses bgp:mp-all-afi-safi-common;
      }
    }

    augment "/rt:routing/rt:control-plane-protocols/" +
            "rt:control-plane-protocol/bgp:bgp/" +
            "bgp:rib/bgp:afi-safis/bgp:afi-safi" {
      when "derived-from-or-self(bgp:name, 'foo')" {
        description
          "This augmentation is valid for a AFI/SAFI instance
           of 'foo'";
      }

      container foo {
        description
          "Container to add 'foo' rib specific information.
           First add the common stuff.";
        container loc-rib {
          config false;
          description
            "Container for the 'foo' BGP LOC-RIB data.";
          container routes {
            description
              "Enclosing container for list of routes in the routing
```

```
               table.";
            list route {
              key "prefix origin path-id";
              description
                "List of routes in the table, keyed by the route
                 prefix, the route origin, and path-id.  The route
                 origin can be either the neighbor address from which
                 the route was learned, or the source protocol that
                 injected the route.  The path-id distinguishes routes
                 for the same prefix received from a neighbor (e.g.,
                 if add-paths is enabled).";
              leaf prefix {
                type inet:ip-address;
                description
                  "The 'foo' prefix corresponding to the route.";
              }
              uses bgp:bgp-loc-rib-common-keys;
              uses bgp:bgp-loc-rib-common-attr-refs;
              uses bgp:bgp-common-route-annotations-state;
              uses bgp:bgp-unknown-attr-top;
              uses bgp:rib-ext-route-annotations;
            }
            uses bgp:clear-routes;
          }
        }

        container neighbors {
          config false;
          description
            "Enclosing container for neighbor list.";
          list neighbor {
            key "neighbor-address";
            description
              "List of neighbors (peers) of the local BGP speaker.";
            leaf neighbor-address {
              type inet:ip-address;
              description
                "IP address of the BGP neighbor or peer.";
            }
            container adj-rib-in-pre {
              description
                "Per-neighbor table containing the NLRI updates
               received from the neighbor before any local input
               policy rules or filters have been applied.  This can
               be considered the 'raw' updates from the neighbor.";
              uses bgp:ipv4-adj-rib-common;
            }
            container adj-rib-in-post {
```

```
              description
                "Per-neighbor table containing the paths received from
               the neighbor that are eligible for best-path selection
               after local input policy rules have been applied.";
              uses bgp:ipv4-adj-rib-in-post;
            }
            container adj-rib-out-pre {
              description
                "Per-neighbor table containing paths eligible for
               sending (advertising) to the neighbor before output
               policy rules have been applied.";
              uses bgp:ipv4-adj-rib-common;
            }
            container adj-rib-out-post {
              description
                "Per-neighbor table containing paths eligible for
               sending (advertising) to the neighbor after output
               policy rules have been applied.";
              uses bgp:ipv4-adj-rib-common;
            }
          }
        }
      }
    }
  }
```

Appendix C. How to deviate a module

This example shows how the BGP can be deviated to indicate two nodes that the particular implementation is choosing not to support.

```
module example-newco-bgp {
  yang-version 1.1;
  namespace "http://example.com/ns/example-newco-bgp";
  prefix example-newco-bgp;

  import ietf-bgp {
    prefix "bgp";
  }

  organization
    "Newco model group.";

  contact
    "abc@newco.com";
  description
    "This YANG module deviates IETF BGP YANG module.";

  revision 2023-07-05 {
    description
      "Creating NewCo deviations to ietf-bgp model";

    reference
      "RFC XXXX: YANG module for Border Gateway Protocol (BGP).";
  }

  deviation "/bgp:bgp/bgp:global/bgp:graceful-restart/" +
            "bgp:restart-time" {
    deviate not-supported;
  }

  deviation "/bgp:bgp/bgp:global/bgp:graceful-restart/" +
            "bgp:stale-route-time" {
    deviate not-supported;
  }
}
```

Appendix D. Complete configuration tree diagram

Here is a complete tree diagram for the configuration and operational part of the model.

```
module: ietf-bgp

  augment /rt:routing/rt:control-plane-protocols
            /rt:control-plane-protocol:
    +--rw bgp
       +--rw global!
       |  +--rw as                          inet:as-number
```

```
          |  +--rw identifier?                yang:dotted-quad
          |  +--rw distance
          |  |  +--rw external?   uint8
          |  |  +--rw internal?   uint8
          |  +--rw confederation
          |  |  +--rw enabled?      boolean
          |  |  +--rw identifier?   inet:as-number
          |  |  +--rw member-as*    inet:as-number
          |  +--rw graceful-restart {bt:graceful-restart}?
          |  |  +--rw enabled?             boolean
          |  |  +--rw restart-time?        bt:graceful-restart-time-type
          |  |  +--rw stale-routes-time?   uint32
          |  |  +--rw helper-only?         boolean
          |  +--rw use-multiple-paths
          |  |  +--rw enabled?   boolean
          |  |  +--rw ebgp
          |  |  |  +--rw allow-multiple-as?   boolean
          |  |  |  +--rw maximum-paths?       uint32
          |  |  +--rw ibgp
          |  |     +--rw maximum-paths?   uint32
          |  +--rw route-selection-options
          |  |  +--rw always-compare-med?           boolean
          |  |  +--rw ignore-as-path-length?        boolean
          |  |  +--rw external-compare-router-id?   boolean
          |  |  +--rw advertise-inactive-routes?    boolean
          |  |  +--rw enable-aigp?                  boolean
          |  |  +--rw ignore-next-hop-igp-metric?   boolean
          |  |  +--rw enable-med?                   boolean
          |  |  +--rw med-plus-igp
          |  |     +--rw enabled?          boolean
          |  |     +--rw igp-multiplier?   uint16
          |  |     +--rw med-multiplier?   uint16
          |  +--rw afi-safis
          |  |  +--rw afi-safi* [name]
          |  |     +--rw name                       identityref
          |  |     +--rw enabled?                   boolean
          |  |     +--ro statistics
          |  |     |  +--ro total-paths?      yang:gauge32
          |  |     |  +--ro total-prefixes?   yang:gauge32
          |  |     +--rw graceful-restart {bt:graceful-restart}?
          |  |     |  +--rw enabled?   boolean
          |  |     +--rw route-selection-options
          |  |     |  +--rw always-compare-med?           boolean
          |  |     |  +--rw ignore-as-path-length?        boolean
          |  |     |  +--rw external-compare-router-id?   boolean
          |  |     |  +--rw advertise-inactive-routes?    boolean
          |  |     |  +--rw enable-aigp?                  boolean
          |  |     |  +--rw ignore-next-hop-igp-metric?   boolean
```

```
|  |      |  +--rw enable-med?                  boolean
|  |      |  +--rw med-plus-igp
|  |      |     +--rw enabled?          boolean
|  |      |     +--rw igp-multiplier?   uint16
|  |      |     +--rw med-multiplier?   uint16
|  |      +--rw add-paths {bt:add-paths}?
|  |      |  +--rw receive?                 boolean
|  |      |  +--rw (send)?
|  |      |  |  +--:(max)
|  |      |  |  |  +--rw max?              uint8
|  |      |  |  +--:(all)
|  |      |  |     +--rw all?              empty
|  |      |  +--rw eligible-prefix-policy?   leafref
|  |      +--rw use-multiple-paths
|  |      |  +--rw enabled?   boolean
|  |      |  +--rw ebgp
|  |      |  |  +--rw allow-multiple-as?   boolean
|  |      |  |  +--rw maximum-paths?       uint32
|  |      |  +--rw ibgp
|  |      |     +--rw maximum-paths?   uint32
|  |      +--rw apply-policy
|  |      |  +--rw import-policy*           leafref
|  |      |  +--rw default-import-policy?   default-policy-type
|  |      |  +--rw export-policy*           leafref
|  |      |  +--rw default-export-policy?   default-policy-type
|  |      +--rw ipv4-unicast
|  |      |  +--rw prefix-limit
|  |      |  |  +--rw max-prefixes?            uint32
|  |      |  |  +--rw warning-threshold-pct?
|  |      |  |  |       rt-types:percentage
|  |      |  |  +--rw teardown?                boolean
|  |      |  |  +--rw idle-time?               union
|  |      |  |  +--ro prefix-limit-exceeded?   boolean
|  |      |  +--rw send-default-route?   boolean
|  |      +--rw ipv6-unicast
|  |      |  +--rw prefix-limit
|  |      |  |  +--rw max-prefixes?            uint32
|  |      |  |  +--rw warning-threshold-pct?
|  |      |  |  |       rt-types:percentage
|  |      |  |  +--rw teardown?                boolean
|  |      |  |  +--rw idle-time?               union
|  |      |  |  +--ro prefix-limit-exceeded?   boolean
|  |      |  +--rw send-default-route?   boolean
|  |      +--rw ipv4-labeled-unicast
|  |      |  +--rw prefix-limit
|  |      |     +--rw max-prefixes?            uint32
|  |      |     +--rw warning-threshold-pct?
|  |      |     |       rt-types:percentage
```

```
|  |      |     +--rw teardown?                boolean
|  |      |     +--rw idle-time?               union
|  |      |     +--ro prefix-limit-exceeded?   boolean
|  |      +--rw ipv6-labeled-unicast
|  |      |  +--rw prefix-limit
|  |      |     +--rw max-prefixes?            uint32
|  |      |     +--rw warning-threshold-pct?
|  |      |     |       rt-types:percentage
|  |      |     +--rw teardown?                boolean
|  |      |     +--rw idle-time?               union
|  |      |     +--ro prefix-limit-exceeded?   boolean
|  |      +--rw l3vpn-ipv4-unicast
|  |      |  +--rw prefix-limit
|  |      |     +--rw max-prefixes?            uint32
|  |      |     +--rw warning-threshold-pct?
|  |      |     |       rt-types:percentage
|  |      |     +--rw teardown?                boolean
|  |      |     +--rw idle-time?               union
|  |      |     +--ro prefix-limit-exceeded?   boolean
|  |      +--rw l3vpn-ipv6-unicast
|  |      |  +--rw prefix-limit
|  |      |     +--rw max-prefixes?            uint32
|  |      |     +--rw warning-threshold-pct?
|  |      |     |       rt-types:percentage
|  |      |     +--rw teardown?                boolean
|  |      |     +--rw idle-time?               union
|  |      |     +--ro prefix-limit-exceeded?   boolean
|  |      +--rw l3vpn-ipv4-multicast
|  |      |  +--rw prefix-limit
|  |      |     +--rw max-prefixes?            uint32
|  |      |     +--rw warning-threshold-pct?
|  |      |     |       rt-types:percentage
|  |      |     +--rw teardown?                boolean
|  |      |     +--rw idle-time?               union
|  |      |     +--ro prefix-limit-exceeded?   boolean
|  |      +--rw l3vpn-ipv6-multicast
|  |      |  +--rw prefix-limit
|  |      |     +--rw max-prefixes?            uint32
|  |      |     +--rw warning-threshold-pct?
|  |      |     |       rt-types:percentage
|  |      |     +--rw teardown?                boolean
|  |      |     +--rw idle-time?               union
|  |      |     +--ro prefix-limit-exceeded?   boolean
|  |      +--rw l2vpn-vpls
|  |      |  +--rw prefix-limit
|  |      |     +--rw max-prefixes?            uint32
|  |      |     +--rw warning-threshold-pct?
|  |      |     |       rt-types:percentage
```

```
          |  |     |     +--rw teardown?                boolean
          |  |     |     +--rw idle-time?               union
          |  |     |     +--ro prefix-limit-exceeded?   boolean
          |  |     +--rw l2vpn-evpn
          |  |        +--rw prefix-limit
          |  |           +--rw max-prefixes?            uint32
          |  |           +--rw warning-threshold-pct?
          |  |           |       rt-types:percentage
          |  |           +--rw teardown?                boolean
          |  |           +--rw idle-time?               union
          |  |           +--ro prefix-limit-exceeded?   boolean
          |  +--rw apply-policy
          |  |  +--rw import-policy*           leafref
          |  |  +--rw default-import-policy?   default-policy-type
          |  |  +--rw export-policy*           leafref
          |  |  +--rw default-export-policy?   default-policy-type
          |  +--ro statistics
          |     +--ro total-paths?      yang:gauge32
          |     +--ro total-prefixes?   yang:gauge32
          +--rw neighbors
          |  +--rw neighbor* [remote-address]
          |  |  +--rw remote-address            inet:ip-address
          |  |  +--rw peer-group?
          |  |  |       -> ../../../peer-groups/peer-group/name
          |  |  +--ro local-address?            inet:ip-address
          |  |  +--ro local-port?               inet:port-number
          |  |  +--ro remote-port?              inet:port-number
          |  |  +--ro peer-type?                bt:peer-type
          |  |  +--ro identifier?               yang:dotted-quad
          |  |  +--ro dynamically-configured?   empty
          |  |  +--rw enabled?                  boolean
          |  |  +--rw peer-as?                  inet:as-number
          |  |  +--rw local-as?                 inet:as-number
          |  |  +--rw remove-private-as?
          |  |  |       bt:remove-private-as-option
          |  |  +--rw route-flap-damping {bt:damping}?
          |  |  |  +--rw enable?           boolean
          |  |  |  +--rw suppress-above?   decimal64
          |  |  |  +--rw reuse-above?      decimal64
          |  |  |  +--rw max-flap?         decimal64
          |  |  |  +--rw reach-decay?      uint32
          |  |  |  +--rw unreach-decay?    uint32
          |  |  |  +--rw keep-history?     uint32
          |  |  +--rw send-community*           identityref
          |  |  |       {bct:send-communities}?
          |  |  +--rw description?              string
          |  |  +--rw timers
          |  |  |  +--rw connect-retry-interval?            uint16
```

```
          |  |  |  +--rw hold-time?                          uint16
          |  |  |  +--ro negotiated-hold-time?               uint16
          |  |  |  +--rw keepalive?                          uint16
          |  |  |  +--rw min-as-origination-interval?        uint16
          |  |  |  +--rw min-route-advertisement-interval?   uint16
          |  |  +--rw transport
          |  |  |  +--rw local-address?    union
          |  |  |  +--rw tcp-mss?          tcp:mss
          |  |  |  +--rw mtu-discovery?    boolean
          |  |  |  +--rw ebgp-multihop
          |  |  |  |  +--rw enabled?        boolean
          |  |  |  |  +--rw multihop-ttl?   uint8
          |  |  |  +--rw passive-mode?     boolean
          |  |  |  +--rw ttl-security?     uint8 {bt:ttl-security}?
          |  |  |  +--rw secure-session
          |  |  |  |  +--rw enabled?   boolean
          |  |  |  |  +--rw options
          |  |  |  |     +--rw (option)?
          |  |  |  |        +--:(ao)
          |  |  |  |        |  +--rw ao-keychain?
          |  |  |  |        |          key-chain:key-chain-ref
          |  |  |  |        +--:(md5)
          |  |  |  |        |  +--rw md5-keychain?
          |  |  |  |        |          key-chain:key-chain-ref
          |  |  |  |        +--:(ipsec)
          |  |  |  |           +--rw sa?             string
          |  |  |  +--rw bfd {bt:bfd}?
          |  |  |     +--rw enabled?                          boolean
          |  |  |     +--rw local-multiplier?                 multiplier
          |  |  |     |       {client-base-cfg-parms}?
          |  |  |     +--rw (interval-config-type)?
          |  |  |             {client-base-cfg-parms}?
          |  |  |        +--:(tx-rx-intervals)
          |  |  |        |  +--rw desired-min-tx-interval?    uint32
          |  |  |        |  +--rw required-min-rx-interval?   uint32
          |  |  |        +--:(single-interval)
          |  |  |                 {single-minimum-interval}?
          |  |  |           +--rw min-interval?               uint32
          |  |  +--rw treat-as-withdraw?        boolean
          |  |  +--rw logging-options
          |  |  |  +--rw log-neighbor-state-changes?   boolean
          |  |  +--rw route-reflector
          |  |  |  +--rw cluster-id?   bt:rr-cluster-id-type
          |  |  |  +--rw client?       boolean
          |  |  +--rw as-path-options
          |  |  |  +--rw allow-own-as?             uint8
          |  |  |  +--rw replace-peer-as?          boolean
          |  |  |  +--rw disable-peer-as-filter?   boolean
```

```
          |  |  +--rw add-paths {bt:add-paths}?
          |  |  |  +--rw receive?                  boolean
          |  |  |  +--rw (send)?
          |  |  |  |  +--:(max)
          |  |  |  |  |  +--rw max?               uint8
          |  |  |  |  +--:(all)
          |  |  |  |     +--rw all?               empty
          |  |  |  +--rw eligible-prefix-policy?   leafref
          |  |  +--rw use-multiple-paths
          |  |  |  +--rw enabled?   boolean
          |  |  |  +--rw ebgp
          |  |  |     +--rw allow-multiple-as?   boolean
          |  |  +--rw apply-policy
          |  |  |  +--rw import-policy*           leafref
          |  |  |  +--rw default-import-policy?   default-policy-type
          |  |  |  +--rw export-policy*           leafref
          |  |  |  +--rw default-export-policy?   default-policy-type
          |  |  +--rw graceful-restart {bt:graceful-restart}?
          |  |  |  +--rw enabled?             boolean
          |  |  |  +--rw restart-time?
          |  |  |  |       bt:graceful-restart-time-type
          |  |  |  +--rw stale-routes-time?   uint32
          |  |  |  +--rw helper-only?         boolean
          |  |  |  +--ro peer-restart-time?   uint16
          |  |  |  +--ro peer-restarting?     boolean
          |  |  |  +--ro local-restarting?    boolean
          |  |  |  +--ro mode?                enumeration
          |  |  +--rw prefix-limit
          |  |  |  +--rw max-prefixes?            uint32
          |  |  |  +--rw warning-threshold-pct?   rt-types:percentage
          |  |  |  +--rw teardown?                boolean
          |  |  |  +--rw idle-time?               union
          |  |  |  +--ro prefix-limit-exceeded?   boolean
          |  |  +--rw afi-safis
          |  |  |  +--rw afi-safi* [name]
          |  |  |     +--rw name                    identityref
          |  |  |     +--rw enabled?                boolean
          |  |  |     +--ro active?                 boolean
          |  |  |     +--ro prefixes
          |  |  |     |  +--ro received?    yang:zero-based-counter32
          |  |  |     |  +--ro sent?        yang:zero-based-counter32
          |  |  |     |  +--ro installed?   yang:gauge32
          |  |  |     +--rw graceful-restart {bt:graceful-restart}?
          |  |  |     |  +--rw enabled?
          |  |  |     |  |       boolean
          |  |  |     |  +--ro received?
          |  |  |     |  |       boolean
          |  |  |     |  +--ro advertised?
```

```
          |  |  |     |  |       boolean
          |  |  |     |  +--ro local-forwarding-state-preserved?
          |  |  |     |  |       boolean
          |  |  |     |  +--ro forwarding-state-preserved?
          |  |  |     |  |       boolean
          |  |  |     |  +--ro end-of-rib-received?
          |  |  |     |          boolean
          |  |  |     +--rw apply-policy
          |  |  |     |  +--rw import-policy*           leafref
          |  |  |     |  +--rw default-import-policy?
          |  |  |     |  |       default-policy-type
          |  |  |     |  +--rw export-policy*           leafref
          |  |  |     |  +--rw default-export-policy?
          |  |  |     |          default-policy-type
          |  |  |     +--rw ipv4-unicast
          |  |  |     |  +--rw prefix-limit
          |  |  |     |  |  +--rw max-prefixes?            uint32
          |  |  |     |  |  +--rw warning-threshold-pct?
          |  |  |     |  |  |       rt-types:percentage
          |  |  |     |  |  +--rw teardown?                boolean
          |  |  |     |  |  +--rw idle-time?               union
          |  |  |     |  |  +--ro prefix-limit-exceeded?   boolean
          |  |  |     |  +--rw send-default-route?   boolean
          |  |  |     +--rw ipv6-unicast
          |  |  |     |  +--rw prefix-limit
          |  |  |     |  |  +--rw max-prefixes?            uint32
          |  |  |     |  |  +--rw warning-threshold-pct?
          |  |  |     |  |  |       rt-types:percentage
          |  |  |     |  |  +--rw teardown?                boolean
          |  |  |     |  |  +--rw idle-time?               union
          |  |  |     |  |  +--ro prefix-limit-exceeded?   boolean
          |  |  |     |  +--rw send-default-route?   boolean
          |  |  |     +--rw ipv4-labeled-unicast
          |  |  |     |  +--rw prefix-limit
          |  |  |     |     +--rw max-prefixes?            uint32
          |  |  |     |     +--rw warning-threshold-pct?
          |  |  |     |     |       rt-types:percentage
          |  |  |     |     +--rw teardown?                boolean
          |  |  |     |     +--rw idle-time?               union
          |  |  |     |     +--ro prefix-limit-exceeded?   boolean
          |  |  |     +--rw ipv6-labeled-unicast
          |  |  |     |  +--rw prefix-limit
          |  |  |     |     +--rw max-prefixes?            uint32
          |  |  |     |     +--rw warning-threshold-pct?
          |  |  |     |     |       rt-types:percentage
          |  |  |     |     +--rw teardown?                boolean
          |  |  |     |     +--rw idle-time?               union
          |  |  |     |     +--ro prefix-limit-exceeded?   boolean
```

```
          |  |  |     +--rw l3vpn-ipv4-unicast
          |  |  |     |  +--rw prefix-limit
          |  |  |     |     +--rw max-prefixes?            uint32
          |  |  |     |     +--rw warning-threshold-pct?
          |  |  |     |     |       rt-types:percentage
          |  |  |     |     +--rw teardown?                boolean
          |  |  |     |     +--rw idle-time?               union
          |  |  |     |     +--ro prefix-limit-exceeded?   boolean
          |  |  |     +--rw l3vpn-ipv6-unicast
          |  |  |     |  +--rw prefix-limit
          |  |  |     |     +--rw max-prefixes?            uint32
          |  |  |     |     +--rw warning-threshold-pct?
          |  |  |     |     |       rt-types:percentage
          |  |  |     |     +--rw teardown?                boolean
          |  |  |     |     +--rw idle-time?               union
          |  |  |     |     +--ro prefix-limit-exceeded?   boolean
          |  |  |     +--rw l3vpn-ipv4-multicast
          |  |  |     |  +--rw prefix-limit
          |  |  |     |     +--rw max-prefixes?            uint32
          |  |  |     |     +--rw warning-threshold-pct?
          |  |  |     |     |       rt-types:percentage
          |  |  |     |     +--rw teardown?                boolean
          |  |  |     |     +--rw idle-time?               union
          |  |  |     |     +--ro prefix-limit-exceeded?   boolean
          |  |  |     +--rw l3vpn-ipv6-multicast
          |  |  |     |  +--rw prefix-limit
          |  |  |     |     +--rw max-prefixes?            uint32
          |  |  |     |     +--rw warning-threshold-pct?
          |  |  |     |     |       rt-types:percentage
          |  |  |     |     +--rw teardown?                boolean
          |  |  |     |     +--rw idle-time?               union
          |  |  |     |     +--ro prefix-limit-exceeded?   boolean
          |  |  |     +--rw l2vpn-vpls
          |  |  |     |  +--rw prefix-limit
          |  |  |     |     +--rw max-prefixes?            uint32
          |  |  |     |     +--rw warning-threshold-pct?
          |  |  |     |     |       rt-types:percentage
          |  |  |     |     +--rw teardown?                boolean
          |  |  |     |     +--rw idle-time?               union
          |  |  |     |     +--ro prefix-limit-exceeded?   boolean
          |  |  |     +--rw l2vpn-evpn
          |  |  |     |  +--rw prefix-limit
          |  |  |     |     +--rw max-prefixes?            uint32
          |  |  |     |     +--rw warning-threshold-pct?
          |  |  |     |     |       rt-types:percentage
          |  |  |     |     +--rw teardown?                boolean
          |  |  |     |     +--rw idle-time?               union
          |  |  |     |     +--ro prefix-limit-exceeded?   boolean
```

```
          |  |  |     +--rw use-multiple-paths
          |  |  |        +--rw enabled?   boolean
          |  |  |        +--rw ebgp
          |  |  |           +--rw allow-multiple-as?   boolean
          |  |  +--rw session-state?            enumeration
          |  |  +--ro last-established?         yang:date-and-time
          |  |  +--ro capabilities
          |  |  |  +--ro advertised-capabilities* [code index]
          |  |  |  |  +--ro code     uint8
          |  |  |  |  +--ro index    uint8
          |  |  |  |  +--ro name?    identityref
          |  |  |  |  +--ro value
          |  |  |  |     +--ro mpbgp
          |  |  |  |     |  +--ro afi?    iana-rt-types:address-family
          |  |  |  |     |  +--ro safi?   iana-rt-types:bgp-safi
          |  |  |  |     |  +--ro name?   identityref
          |  |  |  |     +--ro graceful-restart
          |  |  |  |     |  +--ro flags?
          |  |  |  |     |  |       bt:graceful-restart-flags
          |  |  |  |     |  +--ro unknown-flags?   bits
          |  |  |  |     |  +--ro restart-time?
          |  |  |  |     |  |       bt:graceful-restart-time-type
          |  |  |  |     |  +--ro afi-safis* []
          |  |  |  |     |     +--ro afi?
          |  |  |  |     |     |       iana-rt-types:address-family
          |  |  |  |     |     +--ro safi?
          |  |  |  |     |     |       iana-rt-types:bgp-safi
          |  |  |  |     |     +--ro afi-safi-flags?
          |  |  |  |     |     |       bt:graceful-restart-flags-for-afi
          |  |  |  |     |     +--ro afi-safi-unknown-flags?   bits
          |  |  |  |     +--ro asn32
          |  |  |  |     |  +--ro as?   inet:as-number
          |  |  |  |     +--ro add-paths
          |  |  |  |        +--ro afi-safis* []
          |  |  |  |           +--ro afi?
          |  |  |  |           |       iana-rt-types:address-family
          |  |  |  |           +--ro safi?   iana-rt-types:bgp-safi
          |  |  |  |           +--ro mode?   enumeration
          |  |  |  +--ro received-capabilities* [code index]
          |  |  |  |  +--ro code     uint8
          |  |  |  |  +--ro index    uint8
          |  |  |  |  +--ro name?    identityref
          |  |  |  |  +--ro value
          |  |  |  |     +--ro mpbgp
          |  |  |  |     |  +--ro afi?    iana-rt-types:address-family
          |  |  |  |     |  +--ro safi?   iana-rt-types:bgp-safi
          |  |  |  |     |  +--ro name?   identityref
          |  |  |  |     +--ro graceful-restart
```

```
          |  |  |  |     |  +--ro flags?
          |  |  |  |     |  |       bt:graceful-restart-flags
          |  |  |  |     |  +--ro unknown-flags?   bits
          |  |  |  |     |  +--ro restart-time?
          |  |  |  |     |  |       bt:graceful-restart-time-type
          |  |  |  |     |  +--ro afi-safis* []
          |  |  |  |     |     +--ro afi?
          |  |  |  |     |     |       iana-rt-types:address-family
          |  |  |  |     |     +--ro safi?
          |  |  |  |     |     |       iana-rt-types:bgp-safi
          |  |  |  |     |     +--ro afi-safi-flags?
          |  |  |  |     |     |       bt:graceful-restart-flags-for-afi
          |  |  |  |     |     +--ro afi-safi-unknown-flags?   bits
          |  |  |  |     +--ro asn32
          |  |  |  |     |  +--ro as?   inet:as-number
          |  |  |  |     +--ro add-paths
          |  |  |  |        +--ro afi-safis* []
          |  |  |  |           +--ro afi?
          |  |  |  |           |       iana-rt-types:address-family
          |  |  |  |           +--ro safi?   iana-rt-types:bgp-safi
          |  |  |  |           +--ro mode?   enumeration
          |  |  |  +--ro negotiated-capabilities*   identityref
          |  |  +--ro errors
          |  |  |  +--ro received
          |  |  |  |  +--ro last-notification?
          |  |  |  |  |       yang:date-and-time
          |  |  |  |  +--ro last-error?
          |  |  |  |  |       identityref
          |  |  |  |  +--ro last-error-code?                   uint8
          |  |  |  |  +--ro last-error-subcode?                uint8
          |  |  |  |  +--ro last-encapsulated-error?
          |  |  |  |  |       identityref
          |  |  |  |  +--ro last-encapsulated-error-code?      uint8
          |  |  |  |  +--ro last-encapsulated-error-subcode?   uint8
          |  |  |  |  +--ro last-error-data?                   binary
          |  |  |  +--ro sent
          |  |  |     +--ro last-notification?
          |  |  |     |       yang:date-and-time
          |  |  |     +--ro last-error?
          |  |  |     |       identityref
          |  |  |     +--ro last-error-code?                   uint8
          |  |  |     +--ro last-error-subcode?                uint8
          |  |  |     +--ro last-encapsulated-error?
          |  |  |     |       identityref
          |  |  |     +--ro last-encapsulated-error-code?      uint8
          |  |  |     +--ro last-encapsulated-error-subcode?   uint8
          |  |  |     +--ro last-error-data?                   binary
          |  |  +--ro statistics
```

```
          |  |      +--ro established-transitions?
          |  |      |       yang:zero-based-counter32
          |  |      +--ro messages
          |  |      |  +--ro total-received?
          |  |      |  |       yang:zero-based-counter32
          |  |      |  +--ro total-sent?
          |  |      |  |       yang:zero-based-counter32
          |  |      |  +--ro updates-received?
          |  |      |  |       yang:zero-based-counter32
          |  |      |  +--ro updates-sent?
          |  |      |  |       yang:zero-based-counter32
          |  |      |  +--ro erroneous-updates-withdrawn?
          |  |      |  |       yang:zero-based-counter32
          |  |      |  +--ro erroneous-updates-attribute-discarded?
          |  |      |  |       yang:zero-based-counter32
          |  |      |  +--ro in-update-elapsed-time?
          |  |      |  |       yang:gauge32
          |  |      |  +--ro notifications-received?
          |  |      |  |       yang:zero-based-counter32
          |  |      |  +--ro notifications-sent?
          |  |      |  |       yang:zero-based-counter32
          |  |      |  +--ro route-refreshes-received?
          |  |      |  |       yang:zero-based-counter32
          |  |      |  +--ro route-refreshes-sent?
          |  |      |          yang:zero-based-counter32
          |  |      +--ro queues
          |  |      |  +--ro input?    yang:gauge32
          |  |      |  +--ro output?   yang:gauge32
          |  |      +---x clear {bt:clear-statistics}?
          |  |         +---w input
          |  |         |  +---w clear-at?   yang:date-and-time
          |  |         +--ro output
          |  |            +--ro clear-finished-at?   yang:date-and-time
          |  +---n established
          |  |  +-- remote-address?   -> ../../neighbor/remote-address
          |  +---n backward-transition
          |  |  +-- remote-addr?
          |  |  |       -> ../../neighbor/remote-address
          |  |  +-- notification-received
          |  |  |  +-- last-notification?
          |  |  |  |       yang:date-and-time
          |  |  |  +-- last-error?                        identityref
          |  |  |  +-- last-error-code?                   uint8
          |  |  |  +-- last-error-subcode?                uint8
          |  |  |  +-- last-encapsulated-error?           identityref
          |  |  |  +-- last-encapsulated-error-code?      uint8
          |  |  |  +-- last-encapsulated-error-subcode?   uint8
          |  |  +-- notification-sent
```

```
          |  |      +-- last-notification?
          |  |      |       yang:date-and-time
          |  |      +-- last-error?                        identityref
          |  |      +-- last-error-code?                   uint8
          |  |      +-- last-error-subcode?                uint8
          |  |      +-- last-encapsulated-error?           identityref
          |  |      +-- last-encapsulated-error-code?      uint8
          |  |      +-- last-encapsulated-error-subcode?   uint8
          |  +---x clear {bt:clear-neighbors}?
          |     +---w input
          |     |  +---w (operation)?
          |     |  |  +--:(operation-admin)
          |     |  |  |  +---w admin?          empty
          |     |  |  +--:(operation-hard)
          |     |  |  |  +---w hard?           empty
          |     |  |  +--:(operation-soft)
          |     |  |  |  +---w soft?           empty
          |     |  |  +--:(operation-soft-inbound)
          |     |  |     +---w soft-inbound?   empty {bt:route-refresh}?
          |     |  +---w clear-at?             yang:date-and-time
          |     +--ro output
          |        +--ro clear-finished-at?   yang:date-and-time
          +--rw peer-groups
          |  +--rw peer-group* [name]
          |     +--rw name                  string
          |     +--rw peer-as?              inet:as-number
          |     +--rw local-as?             inet:as-number
          |     +--rw remove-private-as?    bt:remove-private-as-option
          |     +--rw route-flap-damping {bt:damping}?
          |     |  +--rw enable?           boolean
          |     |  +--rw suppress-above?   decimal64
          |     |  +--rw reuse-above?      decimal64
          |     |  +--rw max-flap?         decimal64
          |     |  +--rw reach-decay?      uint32
          |     |  +--rw unreach-decay?    uint32
          |     |  +--rw keep-history?     uint32
          |     +--rw send-community*       identityref
          |     |       {bct:send-communities}?
          |     +--rw description?          string
          |     +--rw timers
          |     |  +--rw connect-retry-interval?             uint16
          |     |  +--rw hold-time?                          uint16
          |     |  +--ro negotiated-hold-time?               uint16
          |     |  +--rw keepalive?                          uint16
          |     |  +--rw min-as-origination-interval?        uint16
          |     |  +--rw min-route-advertisement-interval?   uint16
          |     +--rw transport
          |     |  +--rw local-address?    union
```

```
|      |  +--rw tcp-mss?          tcp:mss
|      |  +--rw mtu-discovery?    boolean
|      |  +--rw ebgp-multihop
|      |  |  +--rw enabled?        boolean
|      |  |  +--rw multihop-ttl?   uint8
|      |  +--rw passive-mode?     boolean
|      |  +--rw ttl-security?     uint8 {bt:ttl-security}?
|      |  +--rw secure-session
|      |  |  +--rw enabled?   boolean
|      |  |  +--rw options
|      |  |     +--rw (option)?
|      |  |        +--:(ao)
|      |  |        |  +--rw ao-keychain?
|      |  |        |          key-chain:key-chain-ref
|      |  |        +--:(md5)
|      |  |        |  +--rw md5-keychain?
|      |  |        |          key-chain:key-chain-ref
|      |  |        +--:(ipsec)
|      |  |           +--rw sa?             string
|      |  +--rw bfd {bt:bfd}?
|      |     +--rw enabled?                          boolean
|      |     +--rw local-multiplier?                 multiplier
|      |     |       {client-base-cfg-parms}?
|      |     +--rw (interval-config-type)?
|      |             {client-base-cfg-parms}?
|      |        +--:(tx-rx-intervals)
|      |        |  +--rw desired-min-tx-interval?    uint32
|      |        |  +--rw required-min-rx-interval?   uint32
|      |        +--:(single-interval)
|      |                 {single-minimum-interval}?
|      |           +--rw min-interval?               uint32
|      +--rw treat-as-withdraw?    boolean
|      +--rw logging-options
|      |  +--rw log-neighbor-state-changes?   boolean
|      +--rw route-reflector
|      |  +--rw cluster-id?   bt:rr-cluster-id-type
|      |  +--rw client?       boolean
|      +--rw as-path-options
|      |  +--rw allow-own-as?             uint8
|      |  +--rw replace-peer-as?          boolean
|      |  +--rw disable-peer-as-filter?   boolean
|      +--rw add-paths {bt:add-paths}?
|      |  +--rw receive?                  boolean
|      |  +--rw (send)?
|      |  |  +--:(max)
|      |  |  |  +--rw max?                uint8
|      |  |  +--:(all)
|      |  |     +--rw all?                empty
```

```
|      |  +--rw eligible-prefix-policy?   leafref
|      +--rw use-multiple-paths
|      |  +--rw enabled?   boolean
|      |  +--rw ebgp
|      |     +--rw allow-multiple-as?   boolean
|      +--rw apply-policy
|      |  +--rw import-policy*           leafref
|      |  +--rw default-import-policy?   default-policy-type
|      |  +--rw export-policy*           leafref
|      |  +--rw default-export-policy?   default-policy-type
|      +--rw dynamic-peers
|      |  +--rw dynamic-peer-list* [prefix]
|      |     +--rw prefix    inet:ip-prefix
|      +--rw graceful-restart {bt:graceful-restart}?
|      |  +--rw enabled?             boolean
|      |  +--rw restart-time?
|      |  |       bt:graceful-restart-time-type
|      |  +--rw stale-routes-time?   uint32
|      |  +--rw helper-only?         boolean
|      +--rw prefix-limit
|      |  +--rw max-prefixes?            uint32
|      |  +--rw warning-threshold-pct?   rt-types:percentage
|      |  +--rw teardown?                boolean
|      |  +--rw idle-time?               union
|      +--rw afi-safis
|         +--rw afi-safi* [name]
|            +--rw name                    identityref
|            +--rw enabled?                boolean
|            +--rw graceful-restart {bt:graceful-restart}?
|            |  +--rw enabled?   boolean
|            +--rw add-paths {bt:add-paths}?
|            |  +--rw receive?                  boolean
|            |  +--rw (send)?
|            |  |  +--:(max)
|            |  |  |  +--rw max?               uint8
|            |  |  +--:(all)
|            |  |     +--rw all?               empty
|            |  +--rw eligible-prefix-policy?   leafref
|            +--rw use-multiple-paths
|            |  +--rw enabled?   boolean
|            |  +--rw ebgp
|            |     +--rw allow-multiple-as?   boolean
|            +--rw apply-policy
|            |  +--rw import-policy*           leafref
|            |  +--rw default-import-policy?
|            |  |       default-policy-type
|            |  +--rw export-policy*           leafref
|            |  +--rw default-export-policy?
```

```
          |           |          default-policy-type
          |           +--rw ipv4-unicast
          |           |  +--rw prefix-limit
          |           |  |  +--rw max-prefixes?            uint32
          |           |  |  +--rw warning-threshold-pct?
          |           |  |  |       rt-types:percentage
          |           |  |  +--rw teardown?                boolean
          |           |  |  +--rw idle-time?               union
          |           |  |  +--ro prefix-limit-exceeded?   boolean
          |           |  +--rw send-default-route?   boolean
          |           +--rw ipv6-unicast
          |           |  +--rw prefix-limit
          |           |  |  +--rw max-prefixes?            uint32
          |           |  |  +--rw warning-threshold-pct?
          |           |  |  |       rt-types:percentage
          |           |  |  +--rw teardown?                boolean
          |           |  |  +--rw idle-time?               union
          |           |  |  +--ro prefix-limit-exceeded?   boolean
          |           |  +--rw send-default-route?   boolean
          |           +--rw ipv4-labeled-unicast
          |           |  +--rw prefix-limit
          |           |     +--rw max-prefixes?            uint32
          |           |     +--rw warning-threshold-pct?
          |           |     |       rt-types:percentage
          |           |     +--rw teardown?                boolean
          |           |     +--rw idle-time?               union
          |           |     +--ro prefix-limit-exceeded?   boolean
          |           +--rw ipv6-labeled-unicast
          |           |  +--rw prefix-limit
          |           |     +--rw max-prefixes?            uint32
          |           |     +--rw warning-threshold-pct?
          |           |     |       rt-types:percentage
          |           |     +--rw teardown?                boolean
          |           |     +--rw idle-time?               union
          |           |     +--ro prefix-limit-exceeded?   boolean
          |           +--rw l3vpn-ipv4-unicast
          |           |  +--rw prefix-limit
          |           |     +--rw max-prefixes?            uint32
          |           |     +--rw warning-threshold-pct?
          |           |     |       rt-types:percentage
          |           |     +--rw teardown?                boolean
          |           |     +--rw idle-time?               union
          |           |     +--ro prefix-limit-exceeded?   boolean
          |           +--rw l3vpn-ipv6-unicast
          |           |  +--rw prefix-limit
          |           |     +--rw max-prefixes?            uint32
          |           |     +--rw warning-threshold-pct?
          |           |     |       rt-types:percentage
```

```
          |           |     +--rw teardown?                boolean
          |           |     +--rw idle-time?               union
          |           |     +--ro prefix-limit-exceeded?   boolean
          |           +--rw l3vpn-ipv4-multicast
          |           |  +--rw prefix-limit
          |           |     +--rw max-prefixes?            uint32
          |           |     +--rw warning-threshold-pct?
          |           |     |       rt-types:percentage
          |           |     +--rw teardown?                boolean
          |           |     +--rw idle-time?               union
          |           |     +--ro prefix-limit-exceeded?   boolean
          |           +--rw l3vpn-ipv6-multicast
          |           |  +--rw prefix-limit
          |           |     +--rw max-prefixes?            uint32
          |           |     +--rw warning-threshold-pct?
          |           |     |       rt-types:percentage
          |           |     +--rw teardown?                boolean
          |           |     +--rw idle-time?               union
          |           |     +--ro prefix-limit-exceeded?   boolean
          |           +--rw l2vpn-vpls
          |           |  +--rw prefix-limit
          |           |     +--rw max-prefixes?            uint32
          |           |     +--rw warning-threshold-pct?
          |           |     |       rt-types:percentage
          |           |     +--rw teardown?                boolean
          |           |     +--rw idle-time?               union
          |           |     +--ro prefix-limit-exceeded?   boolean
          |           +--rw l2vpn-evpn
          |              +--rw prefix-limit
          |                 +--rw max-prefixes?            uint32
          |                 +--rw warning-threshold-pct?
          |                 |       rt-types:percentage
          |                 +--rw teardown?                boolean
          |                 +--rw idle-time?               union
          |                 +--ro prefix-limit-exceeded?   boolean
          +--ro rib
             +--ro attr-sets
             |  +--ro attr-set* [index]
             |     +--ro index         uint64
             |     +--ro attributes
             |        +--ro origin?
             |        |       bt:bgp-origin-attr-type
             |        +--ro as-path
             |        |  +--ro segment* []
             |        |     +--ro type?     identityref
             |        |     +--ro member*   inet:as-number
             |        +--ro next-hop?              inet:ip-address
             |        +--ro link-local-next-hop?   inet:ipv6-address
```

```
             |        +--ro med?                   uint32
             |        +--ro local-pref?            uint32
             |        +--ro as4-path
             |        |  +--ro segment* []
             |        |     +--ro type?     identityref
             |        |     +--ro member*   inet:as-number
             |        +--ro aggregator
             |        |  +--ro as?           inet:as-number
             |        |  +--ro identifier?   yang:dotted-quad
             |        +--ro aggregator4
             |        |  +--ro as4?          inet:as-number
             |        |  +--ro identifier?   yang:dotted-quad
             |        +--ro atomic-aggregate?      boolean
             |        +--ro originator-id?         yang:dotted-quad
             |        +--ro cluster-list*          yang:dotted-quad
             |        +--ro aigp-metric?           uint64
             +--ro communities
             |  +--ro community* [index]
             |     +--ro index        uint64
             |     +--ro community*   union
             +--ro ext-communities
             |  +--ro ext-community* [index]
             |     +--ro index                uint64
             |     +--ro ext-community*       bct:bgp-ext-community-type
             |     +--ro ext-community-raw*   string
             +--ro ipv6-ext-communities
             |  +--ro ipv6-ext-community* [index]
             |     +--ro index                     uint64
             |     +--ro ipv6-ext-community*
             |     |       bct:bgp-ipv6-ext-community-type
             |     +--ro ipv6-ext-community-raw*   string
             +--ro large-communities
             |  +--ro large-community* [index]
             |     +--ro index              uint64
             |     +--ro large-community*   bct:bgp-large-community-type
             +--ro afi-safis
                +--ro afi-safi* [name]
                   +--ro name            identityref
                   +--ro ipv4-unicast
                   |  +--ro loc-rib
                   |  |  +--ro routes
                   |  |     +--ro route* [prefix origin path-id]
                   |  |        +--ro prefix
                   |  |        |       inet:ipv4-prefix
                   |  |        +--ro origin                  union
                   |  |        +--ro path-id                 uint32
                   |  |        +--ro attr-index?             leafref
                   |  |        +--ro community-index?        leafref
```

```
                    |  |        +--ro ext-community-index?     leafref
                    |  |        +--ro large-community-index?   leafref
                    |  |        +--ro last-modified?
                    |  |        |       yang:timeticks
                    |  |        +--ro eligible-route?          boolean
                    |  |        +--ro ineligible-reason?
                    |  |        |       identityref
                    |  |        +--ro unknown-attributes
                    |  |        |  +--ro unknown-attribute* [attr-type]
                    |  |        |     +--ro optional?     boolean
                    |  |        |     +--ro transitive?   boolean
                    |  |        |     +--ro partial?      boolean
                    |  |        |     +--ro extended?     boolean
                    |  |        |     +--ro attr-type     uint8
                    |  |        |     +--ro attr-len?     uint16
                    |  |        |     +--ro attr-value?   binary
                    |  |        +--ro reject-reason?           union
                    |  +--ro neighbors
                    |     +--ro neighbor* [neighbor-address]
                    |        +--ro neighbor-address    inet:ip-address
                    |        +--ro adj-rib-in-pre
                    |        |  +--ro routes
                    |        |  |  +--ro route* [prefix path-id]
                    |        |  |     +--ro prefix
                    |        |  |     |       inet:ipv4-prefix
                    |        |  |     +--ro path-id
                    |        |  |     |       uint32
                    |        |  |     +--ro attr-index?
                    |        |  |     |       leafref
                    |        |  |     +--ro community-index?
                    |        |  |     |       leafref
                    |        |  |     +--ro ext-community-index?
                    |        |  |     |       leafref
                    |        |  |     +--ro large-community-index?
                    |        |  |     |       leafref
                    |        |  |     +--ro last-modified?
                    |        |  |     |       yang:timeticks
                    |        |  |     +--ro eligible-route?
                    |        |  |     |       boolean
                    |        |  |     +--ro ineligible-reason?
                    |        |  |     |       identityref
                    |        |  |     +--ro unknown-attributes
                    |        |  |     |  +--ro unknown-attribute*
                    |        |  |     |          [attr-type]
                    |        |  |     |     +--ro optional?     boolean
                    |        |  |     |     +--ro transitive?   boolean
                    |        |  |     |     +--ro partial?      boolean
                    |        |  |     |     +--ro extended?     boolean
```

```
|         |  |      |     +--ro attr-type     uint8
|         |  |      |     +--ro attr-len?     uint16
|         |  |      |     +--ro attr-value?   binary
|         |  |      +--ro reject-reason?
|         |  |              union
|         |  +--ro clear-routes {bt:clear-routes}?
|         |     +---x clear
|         |        +---w input
|         |        |  +---w clear-at?
|         |        |          yang:date-and-time
|         |        +--ro output
|         |           +--ro clear-finished-at?
|         |                   yang:date-and-time
|         +--ro adj-rib-in-post
|         |  +--ro routes
|         |  |  +--ro route* [prefix path-id]
|         |  |     +--ro prefix
|         |  |     |       inet:ipv4-prefix
|         |  |     +--ro path-id
|         |  |     |       uint32
|         |  |     +--ro attr-index?
|         |  |     |       leafref
|         |  |     +--ro community-index?
|         |  |     |       leafref
|         |  |     +--ro ext-community-index?
|         |  |     |       leafref
|         |  |     +--ro large-community-index?
|         |  |     |       leafref
|         |  |     +--ro last-modified?
|         |  |     |       yang:timeticks
|         |  |     +--ro eligible-route?
|         |  |     |       boolean
|         |  |     +--ro ineligible-reason?
|         |  |     |       identityref
|         |  |     +--ro best-path?
|         |  |     |       boolean
|         |  |     +--ro unknown-attributes
|         |  |     |  +--ro unknown-attribute*
|         |  |     |          [attr-type]
|         |  |     |     +--ro optional?     boolean
|         |  |     |     +--ro transitive?   boolean
|         |  |     |     +--ro partial?      boolean
|         |  |     |     +--ro extended?     boolean
|         |  |     |     +--ro attr-type     uint8
|         |  |     |     +--ro attr-len?     uint16
|         |  |     |     +--ro attr-value?   binary
|         |  |     +--ro reject-reason?
|         |  |              union
```

```
                     |         |  +--ro clear-routes {bt:clear-routes}?
                     |         |     +---x clear
                     |         |        +---w input
                     |         |        |  +---w clear-at?
                     |         |        |          yang:date-and-time
                     |         |        +--ro output
                     |         |           +--ro clear-finished-at?
                     |         |                   yang:date-and-time
                     |         +--ro adj-rib-out-pre
                     |         |  +--ro routes
                     |         |  |  +--ro route* [prefix path-id]
                     |         |  |     +--ro prefix
                     |         |  |     |       inet:ipv4-prefix
                     |         |  |     +--ro path-id
                     |         |  |     |       uint32
                     |         |  |     +--ro attr-index?
                     |         |  |     |       leafref
                     |         |  |     +--ro community-index?
                     |         |  |     |       leafref
                     |         |  |     +--ro ext-community-index?
                     |         |  |     |       leafref
                     |         |  |     +--ro large-community-index?
                     |         |  |     |       leafref
                     |         |  |     +--ro last-modified?
                     |         |  |     |       yang:timeticks
                     |         |  |     +--ro eligible-route?
                     |         |  |     |       boolean
                     |         |  |     +--ro ineligible-reason?
                     |         |  |     |       identityref
                     |         |  |     +--ro unknown-attributes
                     |         |  |     |  +--ro unknown-attribute*
                     |         |  |     |          [attr-type]
                     |         |  |     |     +--ro optional?     boolean
                     |         |  |     |     +--ro transitive?   boolean
                     |         |  |     |     +--ro partial?      boolean
                     |         |  |     |     +--ro extended?     boolean
                     |         |  |     |     +--ro attr-type     uint8
                     |         |  |     |     +--ro attr-len?     uint16
                     |         |  |     |     +--ro attr-value?   binary
                     |         |  |     +--ro reject-reason?
                     |         |  |             union
                     |         |  +--ro clear-routes {bt:clear-routes}?
                     |         |     +---x clear
                     |         |        +---w input
                     |         |        |  +---w clear-at?
                     |         |        |          yang:date-and-time
                     |         |        +--ro output
                     |         |           +--ro clear-finished-at?
```

```
                    |         |                   yang:date-and-time
                    |         +--ro adj-rib-out-post
                    |            +--ro routes
                    |            |  +--ro route* [prefix path-id]
                    |            |     +--ro prefix
                    |            |     |       inet:ipv4-prefix
                    |            |     +--ro path-id
                    |            |     |       uint32
                    |            |     +--ro attr-index?
                    |            |     |       leafref
                    |            |     +--ro community-index?
                    |            |     |       leafref
                    |            |     +--ro ext-community-index?
                    |            |     |       leafref
                    |            |     +--ro large-community-index?
                    |            |     |       leafref
                    |            |     +--ro last-modified?
                    |            |     |       yang:timeticks
                    |            |     +--ro eligible-route?
                    |            |     |       boolean
                    |            |     +--ro ineligible-reason?
                    |            |     |       identityref
                    |            |     +--ro unknown-attributes
                    |            |     |  +--ro unknown-attribute*
                    |            |     |          [attr-type]
                    |            |     |     +--ro optional?     boolean
                    |            |     |     +--ro transitive?   boolean
                    |            |     |     +--ro partial?      boolean
                    |            |     |     +--ro extended?     boolean
                    |            |     |     +--ro attr-type     uint8
                    |            |     |     +--ro attr-len?     uint16
                    |            |     |     +--ro attr-value?   binary
                    |            |     +--ro reject-reason?
                    |            |             union
                    |            +--ro clear-routes {bt:clear-routes}?
                    |               +---x clear
                    |                  +---w input
                    |                  |  +---w clear-at?
                    |                  |          yang:date-and-time
                    |                  +--ro output
                    |                     +--ro clear-finished-at?
                    |                             yang:date-and-time
                    +--ro ipv6-unicast
                       +--ro loc-rib
                       |  +--ro routes
                       |     +--ro route* [prefix origin path-id]
                       |        +--ro prefix
                       |        |       inet:ipv6-prefix
```

```
                        |        +--ro origin                   union
                        |        +--ro path-id                  uint32
                        |        +--ro attr-index?              leafref
                        |        +--ro community-index?         leafref
                        |        +--ro ext-community-index?     leafref
                        |        +--ro large-community-index?   leafref
                        |        +--ro last-modified?
                        |        |       yang:timeticks
                        |        +--ro eligible-route?          boolean
                        |        +--ro ineligible-reason?
                        |        |       identityref
                        |        +--ro unknown-attributes
                        |        |  +--ro unknown-attribute* [attr-type]
                        |        |     +--ro optional?     boolean
                        |        |     +--ro transitive?   boolean
                        |        |     +--ro partial?      boolean
                        |        |     +--ro extended?     boolean
                        |        |     +--ro attr-type     uint8
                        |        |     +--ro attr-len?     uint16
                        |        |     +--ro attr-value?   binary
                        |        +--ro reject-reason?           union
                        +--ro neighbors
                           +--ro neighbor* [neighbor-address]
                              +--ro neighbor-address    inet:ip-address
                              +--ro adj-rib-in-pre
                              |  +--ro routes
                              |  |  +--ro route* [prefix path-id]
                              |  |     +--ro prefix
                              |  |     |       inet:ipv6-prefix
                              |  |     +--ro path-id
                              |  |     |       uint32
                              |  |     +--ro attr-index?
                              |  |     |       leafref
                              |  |     +--ro community-index?
                              |  |     |       leafref
                              |  |     +--ro ext-community-index?
                              |  |     |       leafref
                              |  |     +--ro large-community-index?
                              |  |     |       leafref
                              |  |     +--ro last-modified?
                              |  |     |       yang:timeticks
                              |  |     +--ro eligible-route?
                              |  |     |       boolean
                              |  |     +--ro ineligible-reason?
                              |  |     |       identityref
                              |  |     +--ro unknown-attributes
                              |  |     |  +--ro unknown-attribute*
                              |  |     |          [attr-type]
```

```
                          |  |      |     +--ro optional?     boolean
                          |  |      |     +--ro transitive?   boolean
                          |  |      |     +--ro partial?      boolean
                          |  |      |     +--ro extended?     boolean
                          |  |      |     +--ro attr-type     uint8
                          |  |      |     +--ro attr-len?     uint16
                          |  |      |     +--ro attr-value?   binary
                          |  |      +--ro reject-reason?
                          |  |              union
                          |  +--ro clear-routes {bt:clear-routes}?
                          |     +---x clear
                          |        +---w input
                          |        |  +---w clear-at?
                          |        |          yang:date-and-time
                          |        +--ro output
                          |           +--ro clear-finished-at?
                          |                   yang:date-and-time
                          +--ro adj-rib-in-post
                          |  +--ro routes
                          |  |  +--ro route* [prefix path-id]
                          |  |     +--ro prefix
                          |  |     |       inet:ipv6-prefix
                          |  |     +--ro path-id
                          |  |     |       uint32
                          |  |     +--ro attr-index?
                          |  |     |       leafref
                          |  |     +--ro community-index?
                          |  |     |       leafref
                          |  |     +--ro ext-community-index?
                          |  |     |       leafref
                          |  |     +--ro large-community-index?
                          |  |     |       leafref
                          |  |     +--ro last-modified?
                          |  |     |       yang:timeticks
                          |  |     +--ro eligible-route?
                          |  |     |       boolean
                          |  |     +--ro ineligible-reason?
                          |  |     |       identityref
                          |  |     +--ro best-path?
                          |  |     |       boolean
                          |  |     +--ro unknown-attributes
                          |  |     |  +--ro unknown-attribute*
                          |  |     |          [attr-type]
                          |  |     |     +--ro optional?     boolean
                          |  |     |     +--ro transitive?   boolean
                          |  |     |     +--ro partial?      boolean
                          |  |     |     +--ro extended?     boolean
                          |  |     |     +--ro attr-type     uint8
```

```
|  |      |     +--ro attr-len?     uint16
|  |      |     +--ro attr-value?   binary
|  |      +--ro reject-reason?
|  |             union
|  +--ro clear-routes {bt:clear-routes}?
|     +---x clear
|        +---w input
|        |  +---w clear-at?
|        |          yang:date-and-time
|        +--ro output
|           +--ro clear-finished-at?
|                   yang:date-and-time
+--ro adj-rib-out-pre
|  +--ro routes
|  |  +--ro route* [prefix path-id]
|  |     +--ro prefix
|  |     |       inet:ipv6-prefix
|  |     +--ro path-id
|  |     |       uint32
|  |     +--ro attr-index?
|  |     |       leafref
|  |     +--ro community-index?
|  |     |       leafref
|  |     +--ro ext-community-index?
|  |     |       leafref
|  |     +--ro large-community-index?
|  |     |       leafref
|  |     +--ro last-modified?
|  |     |       yang:timeticks
|  |     +--ro eligible-route?
|  |     |       boolean
|  |     +--ro ineligible-reason?
|  |     |       identityref
|  |     +--ro unknown-attributes
|  |     |  +--ro unknown-attribute*
|  |     |          [attr-type]
|  |     |     +--ro optional?     boolean
|  |     |     +--ro transitive?   boolean
|  |     |     +--ro partial?      boolean
|  |     |     +--ro extended?     boolean
|  |     |     +--ro attr-type     uint8
|  |     |     +--ro attr-len?     uint16
|  |     |     +--ro attr-value?   binary
|  |     +--ro reject-reason?
|  |             union
|  +--ro clear-routes {bt:clear-routes}?
|     +---x clear
|        +---w input
```

```
                              |        |  +---w clear-at?
                              |        |          yang:date-and-time
                              |        +--ro output
                              |           +--ro clear-finished-at?
                              |                   yang:date-and-time
                              +--ro adj-rib-out-post
                                 +--ro routes
                                 |  +--ro route* [prefix path-id]
                                 |     +--ro prefix
                                 |     |       inet:ipv6-prefix
                                 |     +--ro path-id
                                 |     |       uint32
                                 |     +--ro attr-index?
                                 |     |       leafref
                                 |     +--ro community-index?
                                 |     |       leafref
                                 |     +--ro ext-community-index?
                                 |     |       leafref
                                 |     +--ro large-community-index?
                                 |     |       leafref
                                 |     +--ro last-modified?
                                 |     |       yang:timeticks
                                 |     +--ro eligible-route?
                                 |     |       boolean
                                 |     +--ro ineligible-reason?
                                 |     |       identityref
                                 |     +--ro unknown-attributes
                                 |     |  +--ro unknown-attribute*
                                 |     |          [attr-type]
                                 |     |     +--ro optional?     boolean
                                 |     |     +--ro transitive?   boolean
                                 |     |     +--ro partial?      boolean
                                 |     |     +--ro extended?     boolean
                                 |     |     +--ro attr-type     uint8
                                 |     |     +--ro attr-len?     uint16
                                 |     |     +--ro attr-value?   binary
                                 |     +--ro reject-reason?
                                 |             union
                                 +--ro clear-routes {bt:clear-routes}?
                                    +---x clear
                                       +---w input
                                       |  +---w clear-at?
                                       |          yang:date-and-time
                                       +--ro output
                                          +--ro clear-finished-at?
                                                  yang:date-and-time
```

Appendix E. Complete policy tree diagram

Here is a complete tree diagram for the policy portion of the model.

```
module: ietf-bgp-policy

  augment /rt-pol:routing-policy/rt-pol:defined-sets:
    +--rw bgp-defined-sets
       +--rw as-path-sets
       |  +--rw as-path-set* [name]
       |     +--rw name      string
       |     +--rw member*   string
       +--rw community-sets
       |  +--rw community-set* [name]
       |     +--rw name      string
       |     +--rw member*   union
       +--rw ext-community-sets
       |  +--rw ext-community-set* [name]
       |     +--rw name      string
       |     +--rw member*   union
       +--rw ipv6-ext-community-sets
       |  +--rw ipv6-ext-community-set* [name]
       |     +--rw name      string
       |     +--rw member*   union
       +--rw large-community-sets
       |  +--rw large-community-set* [name]
       |     +--rw name      string
       |     +--rw member*   union
       +--rw next-hop-sets
          +--rw next-hop-set* [name]
             +--rw name        string
             +--rw next-hop*   bgp-next-hop-type
  augment /rt-pol:routing-policy/rt-pol:policy-definitions
            /rt-pol:policy-definition/rt-pol:statements
            /rt-pol:statement/rt-pol:conditions:
    +--rw bgp-conditions
       +--rw local-pref
       |  +--rw value?            uint32
       |  +--rw (operation)?
       |     +--:(eq)
       |     |  +--rw eq?         empty
       |     +--:(lt-or-eq)
       |     |  +--rw lt-or-eq?   empty
       |     +--:(gt-or-eq)
       |        +--rw gt-or-eq?   empty
       +--rw med
       |  +--rw value?            uint32
       |  +--rw (operation)?
```

```
          |      +--:(eq)
          |      |  +--rw eq?         empty
          |      +--:(lt-or-eq)
          |      |  +--rw lt-or-eq?   empty
          |      +--:(gt-or-eq)
          |         +--rw gt-or-eq?   empty
          +--rw origin-eq?                      bt:bgp-origin-attr-type
          +--rw match-afi-safi
          |  +--rw afi-safi-in*         identityref
          |  +--rw match-set-options?   match-set-options-type
          +--rw match-neighbor
          |  +--rw neighbor-eq*         inet:ip-address
          |  +--rw match-set-options?   match-set-options-type
          +--rw route-type?                     enumeration
          +--rw community-count
          |  +--rw community-count?   uint32
          |  +--rw (operation)?
          |     +--:(eq)
          |     |  +--rw eq?          empty
          |     +--:(lt-or-eq)
          |     |  +--rw lt-or-eq?    empty
          |     +--:(gt-or-eq)
          |        +--rw gt-or-eq?    empty
          +--rw as-path-length
          |  +--rw as-path-length?   uint32
          |  +--rw (operation)?
          |     +--:(eq)
          |     |  +--rw eq?         empty
          |     +--:(lt-or-eq)
          |     |  +--rw lt-or-eq?   empty
          |     +--:(gt-or-eq)
          |        +--rw gt-or-eq?   empty
          +--rw match-community-set
          |  +--rw community-set?       leafref
          |  +--rw match-set-options?   match-set-options-type
          +--rw match-ext-community-set
          |  +--rw ext-community-set?          leafref
          |  +--rw ext-community-match-kind?   enumeration
          |  +--rw match-set-options?          match-set-options-type
          +--rw match-ipv6-ext-community-set
          |  +--rw ipv6-ext-community-set?          leafref
          |  +--rw ipv6-ext-community-match-kind?   enumeration
          |  +--rw match-set-options?
          |          match-set-options-type
          +--rw match-large-community-set
          |  +--rw large-community-set?   leafref
          |  +--rw match-set-options?     match-set-options-type
          +--rw match-as-path-set
```

```
          |  +--rw as-path-set?         leafref
          |  +--rw match-set-options?   match-set-options-type
          +--rw match-next-hop-set
             +--rw next-hop-set?        leafref
             +--rw match-set-options?   match-set-options-type
     augment /rt-pol:routing-policy/rt-pol:policy-definitions
               /rt-pol:policy-definition/rt-pol:statements
               /rt-pol:statement/rt-pol:actions:
       +--rw bgp-actions
          +--rw set-route-origin?         bt:bgp-origin-attr-type
          +--rw set-local-pref?           uint32
          +--rw set-next-hop?             bgp-next-hop-type
          +--rw set-med?                  bgp-set-med-type
          +--rw set-as-path-prepend
          |  +--rw repeat-n?   uint8
          |  +--rw asn*        inet:as-number
          +--rw set-community
          |  +--rw options?
          |  |       bgp-set-community-option-type
          |  +--rw (method)?
          |     +--:(inline)
          |     |  +--rw communities*         union
          |     +--:(reference)
          |        +--rw community-set-ref?   leafref
          +--rw set-ext-community
          |  +--rw options?
          |  |       bgp-set-community-option-type
          |  +--rw (method)?
          |     +--:(inline)
          |     |  +--rw communities*
          |     |          bct:bgp-ext-community-type
          |     +--:(reference)
          |        +--rw ext-community-set-ref?   leafref
          +--rw set-ipv6-ext-community
          |  +--rw options?
          |  |       bgp-set-community-option-type
          |  +--rw (method)?
          |     +--:(inline)
          |     |  +--rw communities*
          |     |          bct:bgp-ipv6-ext-community-type
          |     +--:(reference)
          |        +--rw ipv6-ext-community-set-ref?   leafref
          +--rw set-large-community
             +--rw options?
             |       bgp-set-community-option-type
             +--rw (method)?
                +--:(inline)
                |  +--rw communities*
```

```
                 |          bct:bgp-large-community-type
                 +--:(reference)
                    +--rw large-community-set-ref?   leafref
```

Appendix F. BGP YANG Model AS_PATH Regular Expressions

F.1. AS_PATH Regular Expressions in Implementations

Implementations of BGP have different ways of rendering the received BGP AS_PATH in their interfaces. Deployed implementations generally render the AS_PATHs consistently from left to right where the left-most AS number matches the neighbor AS; most recently prepended AS, per Section 5.1.2 of [RFC4271]. Deployed implementations also use different characters to match the associated AS_PATH segment types. Those segment types are AS_SEQUENCE, AS_SET, AS_CONFED_SEQUENCE, AS_CONFED_SET. Implementations also have no fully standardized way to render white-space between AS numbers in combination with possible characters denoting the AS_PATH segment types.

This YANG model standardizes no particular rendering format for an AS_PATH. This permits the model to be generally applicable for implementations of BGP and does not conflict with deployed formatting choices.

AS_PATH regular expressions are a common tool used in BGP policy. Such regular expressions match on components of the AS_PATH. In deployed implementations of BGP, there are two common high-level forms of AS_PATH matching:

* Character based tokens. In this form, AS numbers are treated as a string of characters. Regular expressions are evaluated vs. the characters in the string. E.g. "64503+" will match "64503", "645033", etc.

* Integer based tokens. In this form, AS numbers are treated as tokens. Regular expressions are evaluated vs. the full integers. E.g. "64503+" will match "64503", "64503 64503".

F.2. Why not use standard POSIX regular expressions?

* It would be necessary to standardize the format of the AS_PATH in this module. See above for reasons this was not chosen.

* Consistently dealing with separator token in the configured AS_PATH regular expression can be error-prone. Operators interacting with BGP data using "grep" are familiar with this challenge.

* POSIX character-based regular expressions readily map to implementations that match on character based tokens, but do not easily map to integer based token implementation. Integer based token matches can be expressed in implementations that do character based tokens.

* Additionally, YANG by default doesn't use POSIX for its regular expressions.

F.3. BGP YANG AS_PATH Regular Expressions

To address the points above, the BGP YANG AS_PATH regular expressions are a subset of, and are inspired by, the prior work in Section 5.4 of RPSL [RFC2622].

ASN
: ASN is an 4-byte AS Number, from 1..4294967295. 0 is not a valid AS Number and is not permitted to be advertised in BGP AS_PATHs per [RFC7607].

.
: matches any single AS number in the AS_PATH.

[...]
: is an AS number set. It matches AS_PATHs with AS numbers listed between the brackets. The AS numbers in the set are separated by white-space characters. If a '-' is used between two AS numbers in this set, all AS numbers in the range between the two AS numbers are included in the set.

[^...]
: is a complemented AS number set. It matches an AS_PATH which is not matched by the numbers in the set.

^
: Matches the empty string at the beginning of an AS_PATH.

$
: Matches the empty string at the end of an AS_PATH.

Unary postfix operators * + ? {m} {m,n} {m,}
: For a regular expression A, A* matches zero or more occurrences of A; A+ matches one or more occurrences of A; A? matches zero or one occurrence of A; A{m} matches m occurrence of A; A{m,n} matches m to n occurrence of A; A{m,} matches m or more occurrence of A. For example, [AS1 AS2]{2} matches AS1 AS1, AS1 AS2, AS2 AS1, and AS2 AS2.

Binary catenation operator
This is an implicit operator and exists between two regular expressions A and B when no other explicit operator is specified. The resulting expression A B matches an AS-path if A matches some prefix of the AS-path and B matches the rest of the AS-path.

Binary alternative (or) operator |
For a regular expressions A and B, A | B matches any AS-path that is matched by A or B.

(...)
Parenthesis can be used to override the default order of evaluation.

Whitespace in a BGP YANG AS_PATH regular expression is only used to separate tokens within the regular expression. It is otherwise without meaning and may be included or omitted for visual clarity.

F.4. Matching AS_PATH Segment Types in AS_PATH Regular Expressions

There are no deployed implementations of BGP AS_PATH regular expression matches that permit matching of AS_PATH segment types. Known implementations simply ignore the segment types for their internal representation strings used by that implementation's regular expression. While this would seem to be problematic, and may create difficulties in some specific scenarios, the following reasons have mitigated this consideration over the years:

* AS_SET or AS_CONFED_SET in the AS_PATH not only introduces additional character tokens for delimiting the segment, but mean that the contents of that position within the AS_PATH may vary. In general, implementations of AS_SETs will sort the contents of the AS_SET, but this is only a recommendation and not a protocol requirement. AS_SETs are being deprecated for implementation and deployment in BGP due to the issues that they raise in BGP security features. See [I-D.ietf-idr-deprecate-as-set-confed-set].

* BGP Confederation are the always the left-most ASes in the AS_PATH. As a result, matching on confederation ASes may be done using the '^' anchor character against the locally-configured Confederation member ASes. These are typically either publicly registered AS numbers under the control of a single entity, or private AS numbers that are scrubbed as part of route filtering purposes at the provider's network edge.

F.5. BGP YANG AS_PATH Regular Expressions ABNF

TBD...

F.6. Example BGP YANG AS_PATH Regular Expressions

64496
: Matches the AS_PATH that contains at least one instance of 64496, regardless of its position.

^64496$
: Matches the AS_PATH that is exactly one AS long, and that AS is 64496.

^64496
: Matches the AS_PATH where the left-most (neighbor) AS is 64496.

64496$
: Matches the AS_PATH where the right-most (origin) AS is 64496.

[64496 64500]
: Matches the AS_PATH where at least one of the ASes is either 64496 or 64500.

[^64496 64500]
: Matches the AS_PATH where neither 64496 nor 64500 are present in the AS_PATH.

[64496-64511]
: Matches the AS_PATH where at least one AS number is in the range 64496..64511, inclusive. This matches the documentation AS numbers in [RFC5398].

64496 64500
: Matches the AS_PATH where 64496 is present and immediately followed by 64500.

64496 | 64500
: Matches the AS_PATH where 64496 or 64500 occur in the AS_PATH.

^ 64496 .{2} (64500 | [64505-64511])
: Matches the AS_PATH where the left-most (neighbor) AS is 64496, is followed immediately by any two ASes and either 64500 or an AS in the range 64505..64511.

Authors' Addresses

Mahesh Jethanandani
Kloud Services
Email: mjethanandani@gmail.com

Keyur Patel
Arrcus
CA
United States of America
Email: keyur@arrcus.com

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
United States of America
Email: shares@ndzh.com

Jeffrey Haas
Juniper Networks
Email: jhaas@pfrc.org

Inter-Domain Routing
Internet-Draft
Intended status: Standards Track
Expires: May 21, 2020

S. Litkowski
Individual
A. Simpson
Nokia
K. Patel
Arrcus, Inc
J. Haas
Juniper Networks
L. Yong
Huawei

November 18, 2019

Applying BGP flowspec rules on a specific interface set
draft-ietf-idr-flowspec-interfaceset-05

Abstract

The BGP Flow Specification (flowspec) Network Layer Reachability Information (BGP NLRI) extension (draft-ietf-idr-rfc5575bis) is used to distribute traffic flow specifications into BGP. The primary application of this extension is the distribution of traffic filtering policies for the mitigation of distributed denial of service (DDoS) attacks.

By default, flow specification filters are applied on all forwarding interfaces that are enabled for use by the BGP flowspec extension. A network operator may wish to apply a given filter selectively to a subset of interfaces based on an internal classification scheme. Examples of this include "all customer interfaces", "all peer interfaces", "all transit interfaces", etc.

This document defines BGP Extended Communities (RFC4360) that allow such filters to be selectively applied to sets of forwarding interfaces sharing a common group identifier. The BGP Extended Communities carrying this group identifier are referred to as the BGP Flowspec "interface-set" Extended Communities.

Requirements Language

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.

Status of This Memo

Copyright Notice

Table of Contents

1. Use case

While a network may provide connectivity to a homogenous class of users, it often provides connectivity to different groups of users. The nature of these different groups, and how they're classified, varies based on the purpose of the network. In an enterprise network, connectivity may exist between data centers, offices, and external connectivity. In a virtual private networking (VPN) network, it may consist of customers in different sites connected through a VPN, the provider core network, and external networks such as the Internet. In a traditional Internet service provider (ISP) network, the network may consist of points of presence (POPs), internal infrastructure networks, customer networks, peer networks, and transit networks.

The BGP flowspec extension permits traffic filters to be distributed to routers throughout a network. However, these filters often should not be uniformly applied to all network interfaces. As an example, a rate-limiting filter applied to the SMTP protocol may be applied to customer networks, but not other networks. Similarly, a DDoS attack on the SSH protocol may be deemed appropriate to drop at upstream peering routers but not customer routers.

By default, BGP flowspec filters are applied at all interfaces that permit flowspec filters to be installed. What is needed is a way to selectively apply those filters to subsets of interfaces in a network.

2. Interface specific filtering using BGP flowspec

The uses case detailed above require application of different BGP flowspec rules on different sets of interfaces.

We propose to introduce, within BGP flowspec, a traffic filtering scope that identifies a group of interfaces where a particular filter should be applied. Identification of interfaces within BGP flowspec will be done through group identifiers. A group identifier marks a set of interfaces sharing a common administrative property. Like a BGP community, the group identifier itself does not have any significance. It is up to the network administrator to associate a particular meaning to a group identifier value (e.g. group ID#1 associated to Internet customer interfaces). The group identifier is a local interface property. Any interface may be associated with one or more group identifiers using manual configuration.

When a filtering rule advertised through BGP flowspec must be applied only to particular sets of interfaces, the BGP flowspec BGP UPDATE will contain the identifiers associated with the relevant sets of interfaces. In addition to the group identifiers, it will also contain the direction the filtering rule must be applied in (see Section 3).

Configuration of group identifiers associated to interfaces may change over time. An implementation MUST ensure that the filtering rules (learned from BGP flowspec) applied to a particular interface are always updated when the group identifier mapping is changing.

As an example, we can imagine the following design :

- Internet customer interfaces are associated with group-identifier 1.

- VPN customer interfaces are associated with group-identifier 2.

- All customer interfaces are associated with group-identifier 3.

- Peer interfaces are associated with group-identifier 4.

- Transit interfaces are associated with group-identifier 5.

- All external provider interfaces are associated with group-identifier 6.

- All interfaces are associated with group-identifier 7.

If the service provider wants to deploy a specific inbound filtering on external provider interfaces only, the provider can send the BGP flow specification using group-identifier 6 for the inbound direction.

There are some cases where nodes are dedicated to specific functions (Internet peering, Internet Edge, VPN Edge, Service Edge ...), in this kind of scenario, there is an interest for a constrained distribution of filtering rules that are using the interface specific filtering. Without the constrained route distribution, all nodes will received all the filters even if they are not interested in those filters. Constrained route distribution of flowspec filters would allow for a more optimized distribution.

3. Interface-set extended community

This document proposes a new BGP Route Target extended community called the "flowspec interface-set". This document expands the definition of the Route Target extended community to allow a new value of high order octet (Type field) to be 0x07 for the transitive flowspec interface-set extended community, or 0x47 for the non-transitive flowspec interface-set extended community. These are in addition to the values specified in [RFC4360].

This new BGP Route Target extended community is encoded as follows :

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x07 or 0x47  |      0x02     |    Autonomous System Number   :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:     AS Number (cont.)         |O|I|      Group Identifier     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The flags are :

- o O : if set, the flow specification rule MUST be applied in outbound direction to the interface set referenced by the following group-identifier.

- o I : if set, the flow specification rule MUST be applied in inbound direction to the interface set referenced by the following group-identifier.

Both flags can be set at the same time in the interface-set extended community leading to flow rule to be applied in both directions. An interface-set extended community with both flags set to zero MUST be treated as an error and as consequence, the flowspec update MUST be discarded. As having no direction indicated as no sense, there is no need to propagate the filter informations in the network.

The Group Identifier is encoded as a 14-bit number, values 0..16383.

Multiple instances of the interface-set extended community may be present in a BGP update. This may occur if the flowspec rule needs to be applied to multiple sets of interfaces.

Multiple instances of the extended community in a BGP update MUST be interpreted as a "OR" operation. For example, if a BGP UPDATE

contains two interface-set extended communities with group ID 1 and group ID 2, the filter would need to be installed on interfaces belonging to Group ID 1 or Group ID 2.

Similar to using a Route Target extended community, route distribution of flowspec NLRI with interface-set extended communities may be subject to constrained distribution as defined in [RFC4684].

4. Scaling of per-interface rules

In the absence of an interface-set extended community, a flowspec filter is applied to all flowspec enabled interfaces. When interface-set extended communities are present, different interfaces may have different filtering rules, with different terms and actions. These differing rules may make it harder to share forwarding instructions within the forwarding plane.

Flowspec implementations supporting the interface-set extended community SHOULD take care to minimize the scaling impact in such circumstances. How this is accomplished is out of the scope of this document.

5. Deployment Considerations

5.1. Add-Paths

There are some cases where a particular BGP flowspec NLRI may be advertised to different interface groups with a different action. For example, a service provider may want to discard all ICMP traffic from customer interfaces to infrastructure addresses and want to rate-limit the same traffic when it comes from some internal platforms. These particular cases require ADD-PATH ([RFC7911]) to be deployed in order to ensure that all paths (NLRI+interface-set group-id+actions) are propagated within the BGP control plane. Without ADD-PATH, only a single "NLRI+interface-set group-id+actions" will be propagated, so some filtering rules will never be applied.

5.2. Inter-domain Considerations

The Group Identifier used by the interface-set extended community has local significance to its provisioning Autonomous System. While [I-D.ietf-idr-rfc5575bis] permits inter-as advertisement of flowspec NLRI, care must be taken to not accept these communities when they would result in unacceptable filtering policies.

Filtering of interface-set extended communities at Autonomous System border routers (ASBRs) may thus be desirable.

Note that the default behavior without the interface-set feature would to have been to install the flowspec filter on all flowspec enabled interfaces.

6. Security Considerations

This document extends the Security Considerations of [I-D.ietf-idr-rfc5575bis] by permitting flowspec filters to be selectively applied to subsets of network interfaces in a particular direction. Care must be taken to not permit the inadvertant manipulation of the interface-set extended community to bypass expected traffic manipulation.

7. Acknowledgements

Authors would like to thanks Wim Hendrickx and Robert Raszuk for their valuable comments.

8. IANA Considerations

8.1. FlowSpec Transitive Extended Communities

This document requests a new type from the "BGP Transitive Extended Community Types" extended community registry from the First Come First Served range. This type name shall be 'FlowSpec Transitive Extended Communities'. IANA has assigned the value 0x07 to this type.

This document requests creation of a new registry called "FlowSpec Transitive Extended Community Sub-Types". This registry contains values of the second octet (the "Sub-Type" field) of an extended community when the value of the first octet (the "Type" field) is the value allocated in this document. The registration procedure for values in this registry shall be First Come First Served.

8.2. FlowSpec Non-Transitive Extended Communities

This document requests a new type from the "BGP Non-Transitive Extended Community Types" extended community registry from the First Come First Served range. This type name shall be 'FlowSpec Non-Transitive Extended Communities'. IANA has assigned the value 0x47 to this type.

This document requests creation of a new registry called "FlowSpec Non-Transitive Extended Community Sub-Types". This registry contains values of the second octet (the "Sub-Type" field) of an extended community when the value of the first octet (the "Type" field) is the

value allocated in this document. The registration procedure for values in this registry shall be First Come First Served.

8.3. FlowSpec interface-set Extended Community

Within the two new registries above, this document requests a new subtype (suggested value 0x02). This sub-type shall be named "interface-set", with a reference to this document.

8.4. Allocation Advice to IANA

IANA is requested to allocate the values of the FlowSpec Transitive and Non-Transitive Extended Communities such that their values are identical when ignoring the second high-order bit (Transitive). See section 2, [RFC4360].

It is suggested to IANA that, when possible, allocations from the FlowSpec Transitive/Non-Transitive Extended Community Sub-Types registries are made for transitive or non-transitive versions of features (section 2, [RFC4360]) that their code point in both registries is identical.

9. Normative References

[I-D.ietf-idr-rfc5575bis]
Loibl, C., Hares, S., Raszuk, R., McPherson, D., and M. Bacher, "Dissemination of Flow Specification Rules", draft-ietf-idr-rfc5575bis-17 (work in progress), June 2019.

[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>.

[RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, DOI 10.17487/RFC4360, February 2006, <https://www.rfc-editor.org/info/rfc4360>.

[RFC4684] Marques, P., Bonica, R., Fang, L., Martini, L., Raszuk, R., Patel, K., and J. Guichard, "Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)", RFC 4684, DOI 10.17487/RFC4684, November 2006, <https://www.rfc-editor.org/info/rfc4684>.

[RFC7911] Walton, D., Retana, A., Chen, E., and J. Scudder, "Advertisement of Multiple Paths in BGP", RFC 7911, DOI 10.17487/RFC7911, July 2016, <https://www.rfc-editor.org/info/rfc7911>.

[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>.

Authors' Addresses

Stephane Litkowski
Individual

Email: slitkows.ietf@gmail.com

Adam Simpson
Nokia

Email: adam.1.simpson@nokia.com

Keyur Patel
Arrcus, Inc

Email: keyur@arrcus.com

Jeffrey Haas
Juniper Networks

Email: jhaas@juniper.net

Lucy Yong
Huawei

Email: lucy.yong@huawei.com

IDR and SIDR
Internet-Draft
Intended status: Standards Track
Expires: October 20, 2019

K. Sriram, Ed.
USA NIST
A. Azimov, Ed.
Yandex

April 18, 2019

Methods for Detection and Mitigation of BGP Route Leaks
draft-ietf-idr-route-leak-detection-mitigation-11

Abstract

Problem definition for route leaks and enumeration of types of route leaks are provided in RFC 7908. This document describes a solution for detection and mitigation route leaks which is based on conveying route-leak protection (RLP) information in a Border Gateway Protocol (BGP) community. The RLP information is carried in a new well-known transitive BGP community, called the RLP community. The RLP community helps with detection and mitigation of route leaks at ASes downstream from the leaking AS (in the path of the BGP update). This is an inter-AS (multi-hop) solution mechanism. This solution complements the intra-AS (local AS) route-leak avoidance solution that is described in ietf-idr-bgp-open-policy draft.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

RFC 7908 [RFC7908] provides a definition of the route leak problem, and enumerates several types of route leaks. For this document, the definition that is applied is that a route leak occurs when a route received from a transit provider or a lateral peer is forwarded (against commonly used policy) to another transit provider or a lateral peer. The commonly used policy is that a route received from a transit provider or a lateral peer may be forwarded "down only" to customers.

This document describes a solution for detection and mitigation route leaks which is based on conveying route-leak protection (RLP) information in a Border Gateway Protocol (BGP) community. The RLP information is carried in a new well-known transitive BGP community, called the RLP community. The RLP community helps with detection and mitigation of route leaks at ASes downstream from the leaking AS (in the path of the BGP update). This is an inter-AS (multi-hop) solution mechanism. This solution complements the intra-AS (local

AS) route-leak avoidance solution that is described in [I-D.ietf-idr-bgp-open-policy].

Previously, an optional transitive BGP RLP Attribute was proposed to carry the RLP information (in earlier versions of this document). However, this updated document proposes a well-known transitive BGP community to carry the RLP information, with the intention of promoting faster adoption.

The inter-AS RLP mechanism described here can be incrementally deployed. Early adopters would see significant benefits. If a group of big ISPs deploy RLP, then they would be helping each other by blocking route leaks originated within one's customer cone from propagating into a peer's AS or their customer cone.

2. Mechanisms for Detection and Mitigation of Route Leaks

There are two considerations for route leaks: (1) Prevention of route leaks from a local AS [I-D.ietf-idr-bgp-open-policy], and (2) Detection and mitigation of route leaks in ASes that are downstream from the leaking AS (in the path of BGP update). This document specifies the latter.

2.1. Ascertaining Peering Relationship

There are four possible peering relationships (i.e., roles) an AS can have with a neighbor AS: (1) Provider: transit-provider for all prefixes exchanged, (2) Customer: customer for all prefixes exchanged, (3) Lateral Peer: lateral peer (i.e., non-transit) for all prefixes exchanged, and (4) Complex: different relationships for different sets of prefixes [Luckie]. For the complex case, the peering role types provider, customer, or lateral peer apply for different non-overlapping sets of prefixes.

Operators rely on some form of out-of-band (OOB) (i.e., external to BGP) communication to exchange information about their peering relationship, AS number, interface IP address, etc. If the relationship is complex, the OOB communication also includes the sets of prefixes for which they have different roles. [I-D.ietf-idr-bgp-open-policy] introduces a method of re-confirming the BGP Role during BGP OPEN messaging (except when the role is complex). It defines a new BGP Role capability, which helps in re-confirming the relationship when it is provider, customer, or lateral peer. BGP Role does not replace the OOB communication since it relies on the OOB communication to set the role type in the BGP OPEN message. However, BGP Role provides a means to double check, and if there is a contradiction detected via the BGP Role messages, then a Role Mismatch Notification is sent [I-D.ietf-idr-bgp-open-policy].

When the BGP relationship information has been correctly exchanged including the sets of prefixes with different roles (if complex), then this information SHOULD be used to automatically set the role per-prefix with each peer. For example, if the local AS's role is Provider with a neighbor AS, then the per-prefix role is set to 'Provider' for all prefixes sent to the neighbor, and set to 'Customer' for all prefixes received from the neighbor.

Once the per-prefix roles are set, this information is used in the RLP solution mechanism that is described in this document.

2.2. Route-Leak Protection (RLP) Semantics

The key principle is that, in the event of a route leak, a receiving router in a transit-provider AS (e.g., referring to Figure 1, ISP2 (AS2) router) should be able to detect from the RLP community in the update message that its customer AS (e.g., AS3 in Figure 1) should not have forwarded the update (towards the transit-provider AS). Likewise when the update is received from a lateral peer. This means that at least one of the ASes in the AS path of the update put RLP information in RLP community to indicate that it sent the update to its customer or lateral peer, but forbade any subsequent 'Up' (customer to provider) or 'Lateral' (peer to peer) forwarding.

```
                                   /\              /\
                                    \ route-leak(P)/
                                     \ propagated /
                                      \          /
            +------------+    peer    +------------+
      ______| ISP1 (AS1) |----------->|  ISP2 (AS2)|---------->
     /       ------------+  prefix(P) +------------+ route-leak(P)
    | prefix |          \   update      /\        \  propagated
     \  (P)  /           \              /          \
      -------   prefix(P) \            /            \
                   update  \          /              \
                            \        /route-leak(P)  \/
                             \/      /
                        +---------------+
                        | customer(AS3) |
                        +---------------+
```

Figure 1: Illustration of the basic notion of a route leak.

The RLP information contained in the RLP community consists of one or two AS numbers (ASNs) and has the following semantics:

1. Down Only (DO) indication: ASN of the most recent RLP-aware AS in the path to assert that it sent the update to a customer or lateral peer;

2. Leak detected (L) indication: ASN of the first RLP-aware AS in the path to assert that it forwarded the route from a customer or lateral peer despite detecting a leak (to avoid unreachability).

If the RLP community is present in an update, it will always contain a single DO. However, L need not be always present. (Note: The bits designated to carry L may be always present along with a DO, except that a default value (all zeros) is carried in L when no AS in the current AS path needed to assert L.) Once an AS asserts L (Leak detected) by inserting its ASN value, it MUST not be changed subsequently as the update propagates. But the ASN value in DO (Down Only) is changeable along the AS path per its definition above.

Design assumption 1: Operators desire to avoid unreachability. So, a design assumption here is that in the absence of an alternative route, an AS may select and forward a route that is detected to be a leak. (Note: This is the reason Leak detected (L) indication is part of the design.)

Design assumption 2: An AS that is RLP-aware (i.e., implements the RLP solution in this document) MUST also implement an intra-AS solution for route leak avoidance in the local AS. The latter solution uses an intra-AS signaling mechanism (see [I-D.ietf-idr-bgp-open-policy], Section 3.7 of [RLP-Discussion]). By doing this, the AS locally prevents the leaking of routes learned from a transit provider or lateral peer to another transit provider or lateral peer. Why this is critical to the overall solution is made clear in slides 7 and 8 of [sriram2].

2.2.1. Format of the RLP Community

The format of the RLP community using a single Large Community is shown in Figure 2.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Global Administrator  (IANA assigned for RLP)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Local Data Part 1 = DO (ASN value)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Local Data Part 2 = L (ASN value)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Format of the RLP Community using a Large Community [RFC8092].

2.3. Route Leak Detection Rules and the Ingress Router (Receiver) Actions

A received BGP update is determined to be a route leak if:

1. if L is present in the update;

2. else (L is absent), the update is received from a customer and DO is present;

3. else (L is absent), the update is received from a lateral peer and DO is present that is not the lateral peer's ASN.

Note: Here by "L is present" we mean that its value is not the default value (all zeros) but is a proper ASN. Effectively "L is absent" if its value is the default value.

In steps 2 and 3 above, the ingress router (receiver) MUST add L = local ANS. Doing this prior to the best path selection process is necessary. Also, if the route is selected as best path, then L is already set correctly before the egress router (sender) acts on it.

2.4. Route Selection Policy

Minimum Default Policy: Whenever there is a choice between a customer route and a provider route that are both detected to be leaks (L is present), then lower the LocalPref to X (TBD by operator) for each of them. Then shortest path criterion would typically make the customer route preferred. (Note: This would help mitigate any possibility of persistent oscillation; see slide #7 in [sriram1].)

Generalized Minimum Default Policy: Whenever there is a choice between multiple routes (customer/peer/provider) and each is detected to be a leak (L is present), then lower the LocalPref to X TBD by

operator) for each of them. Then apply shortest path criterion. (Note: Some network operators may find this inadequate; see scenarios #3 and #6 in slides #14 and #16, respectively, in [sriram2]. But they may locally modify their policy while respecting the basic principle.)

2.5. Egress Router (Sender) Actions

After best path selection has been performed, a sender MUST perform the following RLP-related actions on the update to be propagated:

1. When propagating a route originated by the local AS to a customer or lateral peer, add DO = local ASN;

2. Else, when propagating a route that already includes a DO (i.e., was received with a DO) to a customer or lateral peer, replace the DO value with the local ASN.

3. Pseudo Code

[Begin: receiver action for route leak detection]

{Comment: This precedes route selection policy.}

```
if received route includes L, then save the route in RIB-in as is;

else (L is absent), if route is received from a customer and DO is
preset, then add L = local ASN;

else (L is absent), if route is received from a lateral peer and
DO is present that is not the lateral peer's ASN, then add L =
local ASN
```

{Comment: "Route does not include L" or "L is absent" only if L is either literally absent or has the default (all zeros) value.}

[End: receiver action for route leak detection]

[Begin: route selection policy]

```
for each route that includes L, lower the LocalPref to X (TBD);
apply best path selection policy*
```

{*Comment: E.g., best path selection based on LocalPref first and then shortest path.}

[End: route selection policy]

[Begin: sender action]

{Comment: RLP (includes DO and L or just DO) is a *transitive* BGP community and should propagate globally.}

when propagating a route originated by local AS to a customer or lateral peer, add DO = local ASN;

when propagating a route that includes a DO (i.e., was received with a DO) to a customer or lateral peer, replace the DO value with the local ASN;

[End: sender action]

4. Security Considerations

With the use of BGP community, there is often a concern that the community propagates beyond its intended perimeter and causes harm [streibelt]. However, that concern does not apply to the RLP community because it is a transitive community that must propagate as far as the update goes.

The proposed Route-Leak Protection (RLP) information carried in the RLP community can benefit from cryptographic protection to prevent abuse by malicious actors in the AS path. In the future, if there is BGPsec deployment, the RLP information can be encoded in the Flags field in the Secure_Path Segment in BGPsec updates [RFC8205]. So, the cryptographic security mechanisms in BGPsec can also secure the RLP information. The reader is directed to the security considerations provided in [RFC8205].

5. IANA Considerations

IANA is requested to register RLP in the well-known Large Community [RFC8092] registry (need help to clarify this). IANA is requested to allocate a new Global Administrator ID for the RLP community (Large Community) (see Figure 2 in this document). Note that BGP Path Attribute value for Large Community is 32 (IANA allocated) [RFC8092].

6. References

6.1. Normative References

[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>.

6.2. Informative References

[draft-dickson-sidr-route-leak-solns] Dickson, B., "Route Leaks -- Proposed Solutions", IETF Internet Draft (expired), March 2012, <https://tools.ietf.org/html/draft-dickson-sidr-route-leak-solns-01>.

[I-D.ietf-idr-bgp-open-policy] Azimov, A., Bogomazov, E., Bush, R., Patel, K., and K. Sriram, "Route Leak Prevention using Roles in Update and Open messages", draft-ietf-idr-bgp-open-policy-05 (work in progress), February 2019.

[Luckie] Luckie, M., Huffaker, B., Dhamdhere, A., Giotsas, V., and kc. claffy, "AS Relationships, Customer Cones, and Validation", IMC 2013, October 2013, <http://www.caida.org/~amogh/papers/asrank-IMC13.pdf>.

[RFC6811] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, "BGP Prefix Origin Validation", RFC 6811, DOI 10.17487/RFC6811, January 2013, <https://www.rfc-editor.org/info/rfc6811>.

[RFC7454] Durand, J., Pepelnjak, I., and G. Doering, "BGP Operations and Security", BCP 194, RFC 7454, DOI 10.17487/RFC7454, February 2015, <https://www.rfc-editor.org/info/rfc7454>.

[RFC7908] Sriram, K., Montgomery, D., McPherson, D., Osterweil, E., and B. Dickson, "Problem Definition and Classification of BGP Route Leaks", RFC 7908, DOI 10.17487/RFC7908, June 2016, <https://www.rfc-editor.org/info/rfc7908>.

[RFC8092] Heitz, J., Ed., Snijders, J., Ed., Patel, K., Bagdonas, I., and N. Hilliard, "BGP Large Communities Attribute", RFC 8092, DOI 10.17487/RFC8092, February 2017, <https://www.rfc-editor.org/info/rfc8092>.

[RFC8205] Lepinski, M., Ed. and K. Sriram, Ed., "BGPsec Protocol Specification", RFC 8205, DOI 10.17487/RFC8205, September 2017, <https://www.rfc-editor.org/info/rfc8205>.

[RLP-Discussion]
Sriram (Ed.), K., "Design Discussion of Route Leaks Solution Methods", Work in Progress, draft-sriram-idr-route-leak-solution-discussion-00, July 2018, <https://tools.ietf.org/html/draft-sriram-idr-route-leak-solution-discussion-00>.

[sriram1] Sriram et al., K., "Route Leaks Solution Merger of RLP and eOTC Drafts", Presented at the IDR Working Group Meeting, IETF-102, Montreal, July 2018, <https://datatracker.ietf.org/meeting/102/materials/slides-102-idr-sessb-route-leaks-merged-solution-00>.

[sriram2] Sriram et al., K., "Solution for Route Leaks Using BGP Communities", Authors Team Discussion Slides, October 2018, <https://www.nist.gov/sites/default/files/documents/2018/10/22/rlp_using_bgp_community-v4.pdf>.

[streibelt]
Streibelt et al., F., "BGP Communities: Even more Worms in the Routing Can", ACM IMC, October 2018, <https://archive.psg.com//181101.imc-communities.pdf>.

Acknowledgements

The authors wish to thank John Scudder and Susan Hares for their review and comments.

Contributors

The following people made significant contributions to this document and should be considered co-authors:

Brian Dickson
Independent
Email: brian.peter.dickson@gmail.com

Doug Montgomery
USA National Institute of Standards and Technology
Email: dougm@nist.gov

Keyur Patel
Arrcus
Email: keyur@arrcus.com

Andrei Robachevsky
Internet Society
Email: robachevsky@isoc.org

Eugene Bogomazov
Qrator Labs
Email: eb@qrator.net

Randy Bush
Internet Initiative Japan
Email: randy@psg.com

Authors' Addresses

Kotikalapudi Sriram (editor)
USA National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, MD 20899
United States of America

Email: ksriram@nist.gov

Alexander Azimov (editor)
Yandex
Moscow
Russia

Email: a.e.azimov@gmail.com

Network Working Group
Internet-Draft
Updates: 9012 (if approved)
Intended status: Standards Track
Expires: 25 April 2024

S. Previdi
Huawei Technologies
C. Filsfils
K. Talaulikar, Ed.
Cisco Systems
P. Mattes
Microsoft
D. Jain
Google

23 October 2023

Advertising Segment Routing Policies in BGP
draft-ietf-idr-segment-routing-te-policy-26

Abstract

This document introduces a BGP SAFI with two NLRIs to advertise a candidate path of a Segment Routing (SR) Policy. An SR Policy is an ordered list of segments (i.e., instructions) that represent a source-routed policy. An SR Policy consists of one or more candidate paths, each consisting of one or more segment lists. A headend may be provisioned with candidate paths for an SR Policy via several different mechanisms, e.g., CLI, NETCONF, PCEP, or BGP. This document specifies how BGP may be used to distribute SR Policy candidate paths. It defines sub-TLVs for the Tunnel Encapsulation Attribute for signaling information about these candidate paths.

This documents updates RFC9012 with extensions to the Color Extended Community to support additional steering modes over SR Policy.

Copyright Notice

Table of Contents

1. Introduction

Segment Routing (SR) [RFC8402] allows a headend node to steer a packet flow along a specific path. Intermediate per-path states are eliminated thanks to source routing.

The headend node is said to steer a flow into an SR Policy [RFC8402].

The packets steered into an SR Policy carry an ordered list of segments associated with that SR Policy.

[RFC9256] further details the concepts of SR Policy and steering into an SR Policy. These apply equally to the SR-MPLS and Segment Routing for IPv6 (SRv6) data-plane instantiations of Segment Routing using SR-MPLS and SRv6 Segment Identifiers (SIDs) as described in [RFC8402]. [RFC8660] describes the representation and processing of this ordered list of segments as an MPLS label stack for SR-MPLS. While [RFC8754] and [RFC8986] describe the same for SRv6 with the use of the Segment Routing Header (SRH).

The SR Policy related functionality described in [RFC9256] can be conceptually viewed as being incorporated in an SR Policy Module (SRPM). Following is a reminder of the high-level functionality of SRPM:

* Learning multiple candidate paths for an SR Policy via various mechanisms (CLI, NETCONF, PCEP, or BGP).

* Selection of the best candidate path for an SR Policy.

* Associating a Binding SID (BSID) to the selected candidate path of an SR Policy.

* Installation of the selected candidate path and its BSID in the forwarding plane.

This document specifies the use of BGP to distribute one or more of the candidate paths of an SR Policy to the headend of that policy. The document describes the functionality provided by BGP and, as appropriate, provides references for the functionality which is outside the scope of BGP (i.e. resides within SRPM on the headend node).

This document specifies a way of representing SR Policy candidate paths in BGP UPDATE messages. BGP can then be used to propagate the SR Policy candidate paths to the headend nodes in a network. The usual BGP rules for BGP propagation and best-path selection are used. At the headend of a specific policy, this will result in one or more candidate paths being installed into the "BGP table". These paths are then passed to the SRPM. The SRPM may compare them to candidate paths learned via other mechanisms and will choose one or more paths to be installed in the data plane. BGP itself does not install SR Policy candidate paths into the data plane.

This document introduces a BGP subsequent address family (SAFI) for IPv4 and IPv6 address families. In UPDATE messages of those AFI/SAFIs, the NLRI identifies an SR Policy Candidate Path while the attributes encode the segment lists and other details of that SR Policy Candidate Path.

While for simplicity we may write that BGP advertises an SR Policy, it has to be understood that BGP advertises a candidate path of an SR policy and that this SR Policy might have several other candidate paths provided via BGP (via an NLRI with a different distinguisher as defined in Section 2.1), PCEP, NETCONF, or local policy configuration.

Typically, a controller defines the set of policies and advertises them to policy headend routers (typically ingress routers). These policy advertisements use the BGP extensions defined in this document. The policy advertisement is, in most but not all cases, tailored for a specific policy headend. In this case, the advertisement may be sent on a BGP session to that headend and not propagated any further.

Alternatively, a router (i.e., a BGP egress router) advertises SR Policies representing paths to itself. In this case, it is possible to send the policy to each headend over a BGP session to that headend, without requiring any further propagation of the policy.

An SR Policy intended only for the receiver will, in most cases, not traverse any Route Reflector (RR, [RFC4456]).

In some situations, it is undesirable for a controller or BGP egress router to have a BGP session to each policy headend. In these situations, BGP Route Reflectors may be used to propagate the advertisements. In certain other deployments, it may be necessary for the advertisement to propagate through a sequence of one or more ASes within an SR Domain (refer to Section 7 for the associated security considerations). To make this possible, an attribute needs to be attached to the advertisement that enables a BGP speaker to determine whether it is intended to be a headend for the advertised policy. This is done by attaching one or more Route Target Extended Communities to the advertisement [RFC4360].

The BGP extensions for the advertisement of SR Policies include following components:

* A Subsequent Address Family Identifier (SAFI) whose NLRIs identifies an SR Policy candidate path.

* A Tunnel Type identifier for SR Policy, and a set of sub-TLVs to be inserted into the Tunnel Encapsulation Attribute (as defined in [RFC9012]) specifying segment lists of the SR Policy candidate path, as well as other information about the SR Policy.

* One or more IPv4 address format route target extended community ([RFC4360]) attached to the SR Policy advertisement and that indicates the intended headend of such an SR Policy advertisement.

The Color Extended Community (as defined in [RFC9012]) is used to steer traffic into an SR Policy, as described in section 8.8 of [RFC9256]. The Section 3 of this document updates [RFC9012] with modifications to the format of the Flags field of the Color Extended Community by using the two leftmost bits of that field.

1.1. Requirements Language

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.

2. SR Policy Encoding

2.1. SR Policy SAFI and NLRI

A SAFI is introduced in this document: the SR Policy SAFI with codepoint 73. The AFI used MUST be IPv4(1) or IPv6(2).

The SR Policy SAFI uses the NLRI format defined as follows:

```
+------------------+
|  NLRI Length     | 1 octet
+------------------+
|  Distinguisher   | 4 octets
+------------------+
|  Policy Color    | 4 octets
+------------------+
|  Endpoint        | 4 or 16 octets
+------------------+
```

Figure 1: SR Policy SAFI Format

where:

* NLRI Length: 1 octet indicating the length expressed in bits as defined in [RFC4760]. When AFI = 1 the value MUST be 96 and when AFI = 2 the value MUST be 192.

* Distinguisher: 4-octet value uniquely identifying the policy in the context of <color, endpoint> tuple. The distinguisher has no semantic value and is solely used by the SR Policy originator to make unique (from an NLRI perspective) both for multiple candidate paths of the same SR Policy as well as candidate paths of different SR Policies (i.e. with different segment lists) with the same Color and Endpoint but meant for different headends.

* Policy Color: 4-octet value identifying (with the endpoint) the policy. The color is used to match the color of the destination prefixes to steer traffic into the SR Policy as specified in section 8 of [RFC9256].

* Endpoint: value identifies the endpoint of a policy. The Endpoint may represent a single node or a set of nodes (e.g., an anycast address). The Endpoint is an IPv4 (4-octet) address or an IPv6 (16-octet) address according to the AFI of the NLRI. The address can be either a unicast or an unspecified address (0.0.0.0 for IPv4, :: for IPv6) as specified in section 2.1 of [RFC9256].

The color and endpoint are used to automate the steering of BGP service routes on SR Policy as described in section 8 of [RFC9256].

The NLRI containing an SR Policy candidate path is carried in a BGP UPDATE message [RFC4271] using BGP multi-protocol extensions [RFC4760] with an AFI of 1 or 2 (IPv4 or IPv6) and with a SAFI of 73. The fault management and error handling in the encoding of the NLRI is specified in Section 5.

An update message that carries the MP_REACH_NLRI or MP_UNREACH_NLRI attribute with the SR Policy SAFI MUST also carry the BGP mandatory attributes. In addition, the BGP update message MAY also contain any of the BGP optional attributes.

The next-hop network address field in SR Policy SAFI (73) updates may be either a 4-octet IPv4 address or a 16-octet IPv6 address, independent of the SR Policy AFI. The length field of the next-hop address specifies the next-hop address family. If the next-hop length is 4, then the next-hop is an IPv4 address; if the next-hop length is 16, then it is a global IPv6 address; if the next-hop length is 32, then it has a global IPv6 address followed by a link-local IPv6 address. The setting of the next-hop field and its attendant processing is governed by standard BGP procedures as described in section 3 of [RFC4760] and section 3 of [RFC2545].

It is important to note that any BGP speaker receiving a BGP message with an SR Policy NLRI, will process it only if the NLRI is among the best paths as per the BGP best-path selection algorithm. In other words, this document leverages the existing BGP propagation and best-path selection rules. Details of the procedures are described in Section 4.

It has to be noted that if several candidate paths of the same SR Policy (endpoint, color) are signaled via BGP to a headend, then it is RECOMMENDED that each NLRI uses a different distinguisher. If BGP has installed into the BGP table two advertisements whose respective NLRIs have the same color and endpoint, but different distinguishers, both advertisements are passed to the SRPM as different candidate paths along with their respective originator information (i.e., ASN and BGP Router-ID) as described in section 2.4 of [RFC9256]. The ASN would be the ASN of the origin and the BGP Router-ID is determined in the following order:

* From the Route Origin Community [RFC4360] if present and carrying an IP Address, or

* As the BGP Originator ID [RFC4456] if present, or

* As the BGP Router-ID of the peer from which the update was received as a last resort.

2.2. SR Policy and Tunnel Encapsulation Attribute

The content of the SR Policy Candidate Path is encoded in the Tunnel Encapsulation Attribute defined in [RFC9012] using a Tunnel-Type called SR Policy Type with codepoint 15. The use of SR Policy Tunnel-type is applicable only for the AFI/SAFI pairs of (1/73, 2/73).

The SR Policy Encoding structure is as follows:

```
SR Policy SAFI NLRI: <Distinguisher, Policy-Color, Endpoint>
Attributes:
   Tunnel Encapsulation Attribute (23)
      Tunnel Type: SR Policy (15)
          Binding SID
          SRv6 Binding SID
          Preference
          Priority
          Policy Name
          Policy Candidate Path Name
          Explicit NULL Label Policy (ENLP)
          Segment List
              Weight
              Segment
              Segment
              ...
          ...
```

Figure 2: SR Policy Encoding

where:

* SR Policy SAFI NLRI is defined in Section 2.1.

* Tunnel Encapsulation Attribute is defined in [RFC9012].

* Tunnel-Type is set to 15.

* Preference, Binding SID, SRv6 Binding SID, Priority, Policy Name, Policy Candidate Path Name, ENLP, Segment-List, Weight, and Segment sub-TLVs are defined in Section 2.4.

* Additional sub-TLVs may be defined in the future.

A Tunnel Encapsulation Attribute MUST NOT contain more than one TLV of type "SR Policy".

2.3. Remote Endpoint and Color

The Tunnel Egress Endpoint and Color sub-TLVs, as defined in [RFC9012], may also be present in the SR Policy encodings.

The Tunnel Egress Endpoint and Color Sub-TLVs of the Tunnel Encapsulation Attribute are not used for SR Policy encodings and therefore their value is irrelevant in the context of the SR Policy SAFI NLRI. If present, the Tunnel Egress Endpoint sub-TLV and the Color sub-TLV MUST be ignored by the BGP speaker and MUST NOT be removed from the Tunnel Encapsulation Attribute during propagation.

2.4. SR Policy Sub-TLVs

This section specifies the sub-TLVs defined for encoding the information about the SR Policy Candidate Path.

Preference, Binding SID, SRv6 Binding SID, Segment-List, Priority, Policy Name, Policy Candidate Path Name, and Explicit NULL Label Policy are the sub-TLVs introduced for the BGP Tunnel Encapsulation Attribute [RFC9012] being defined in this section.

Weight and Segment are sub-TLVs of the Segment-List sub-TLV mentioned above.

The fault management and error handling in the encoding of the sub-TLVs defined in this section are specified in Section 5.

None of the sub-TLVs defined in the following sub-sections have any effect on the BGP best-path selection or propagation procedures. These sub-TLVs are not used by the BGP path selection process and are instead passed on to SRPM as SR Policy Candidate Path information for further processing described in section 2 of [RFC9256].

The use of SR Policy Sub-TLVs is applicable only for the AFI/SAFI pairs of (1/73, 2/73). Future documents may extend their applicability to other AFI/SAFI.

2.4.1. Preference Sub-TLV

The Preference sub-TLV is used to carry the Preference of an SR Policy candidate path. The contents of this sub-TLV are used by the SRPM as described in section 2.7 of [RFC9256].

The Preference sub-TLV is optional and it MUST NOT appear more than once in the SR Policy encoding.

The Preference sub-TLV has following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Length      |     Flags     |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Preference (4 octets)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Preference sub-TLV

where:

* Type: 12

* Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value MUST be 6.

* Flags: 1 octet of flags. No flags are defined in this document. The Flags field MUST be set to zero on transmission and MUST be ignored on receipt.

* RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

* Preference: a 4-octet value indicating the Preference of the SR Policy Candidate Path as described in section 2.7 of [RFC9256].

2.4.2. Binding SID Sub-TLV

The Binding SID sub-TLV is used to signal the binding SID related information of the SR Policy candidate path. The contents of this sub-TLV are used by the SRPM as described in section 6 in [RFC9256].

The Binding SID sub-TLV is optional and it MUST NOT appear more than once in the SR Policy encoding.

When the Binding SID sub-TLV is used to signal an SRv6 SID, the choice of its SRv6 Endpoint Behavior [RFC8986] to be instantiated is left to the headend node. It is RECOMMENDED that the SRv6 Binding SID sub-TLV defined in Section 2.4.3, that enables the specification of the SRv6 Endpoint Behavior, be used for signaling of an SRv6 Binding SID for an SR Policy candidate path.

The Binding SID sub-TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Length      |     Flags     |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Binding SID (variable, optional)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Binding SID sub-TLV

where:

* Type: 13

* Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value MUST be one of: 18 when a SRv6 BSID is present, 6 when a SR-MPLS BSID is present, or 2 when no BSID is present.

* Flags: 1 octet of flags. The following flags are defined in the registry "SR Policy Binding SID Flags" as described in Section 6.6:

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|S|I|           |
+-+-+-+-+-+-+-+-+
```

  Figure 5: Binding SID Flags

  where:

  - S-Flag: This flag encodes the "Specified-BSID-only" behavior. It is used by SRPM as described in section 6.2.3 in [RFC9256].

  - I-Flag: This flag encodes the "Drop Upon Invalid" behavior. It is used by SRPM as described in section 8.2 in [RFC9256].

  - The unassigned bits in the Flag octet MUST be set to zero upon transmission and MUST be ignored upon receipt.

* RESERVED: 1 octet of reserved bits. MUST be set to zero on transmission and MUST be ignored on receipt.

* Binding SID: If the length is 2, then no Binding SID is present. If the length is 6 then the Binding SID is encoded in 4 octets using the format below. Traffic Class (TC), S, and TTL (Total of 12 bits) are RESERVED and MUST be set to zero and MUST be ignored.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Label                        | TC  |S|       TTL     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Binding SID Label Encoding

If the length is 18 then the Binding SID contains a 16-octet SRv6 SID.

2.4.3. SRv6 Binding SID Sub-TLV

The SRv6 Binding SID sub-TLV is used to signal the SRv6 Binding SID related information of an SR Policy candidate path. It enables the specification of the SRv6 Endpoint Behavior [RFC8986] to be instantiated on the headend node. The contents of this sub-TLV are used by the SRPM as described in section 6 in [RFC9256].

The SRv6 Binding SID sub-TLV is optional. More than one SRv6 Binding SID sub-TLVs MAY be signaled in the same SR Policy encoding to indicate one or more SRv6 SIDs, each with potentially different SRv6 Endpoint Behaviors to be instantiated.

The SRv6 Binding SID sub-TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Length      |     Flags     |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 SRv6 Binding SID (16 octets)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//     SRv6 Endpoint Behavior and SID Structure (optional)     //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: SRv6 Binding SID sub-TLV

where:

* Type: 20

* Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value MUST be 26 when the SRv6 Endpoint Behavior and SID Structure is present else it MUST be 18.

* Flags: 1 octet of flags. The following flags are defined in the registry "SR Policy Binding SID Flags" as described in Section 6.7:

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|S|I|B|         |
+-+-+-+-+-+-+-+-+
```

Figure 8: SRv6 Binding SID Flags

where:

- S-Flag: This flag encodes the "Specified-BSID-only" behavior. It is used by SRPM as described in section 6.2.3 in [RFC9256].

- I-Flag: This flag encodes the "Drop Upon Invalid" behavior. It is used by SRPM as described in section 8.2 in [RFC9256].

- B-Flag: This flag, when set, indicates the presence of the SRv6 Endpoint Behavior and SID Structure encoding specified in Section 2.4.4.2.4.

- The unassigned bits in the Flag octet MUST be set to zero upon transmission and MUST be ignored upon receipt.

* RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

* SRv6 Binding SID: Contains a 16-octet SRv6 SID.

* SRv6 Endpoint Behavior and SID Structure: Optional, as defined in Section 2.4.4.2.4.

2.4.4. Segment List Sub-TLV

The Segment List sub-TLV encodes a single explicit path towards the endpoint as described in section 5.1 of [RFC9256]. The Segment List sub-TLV includes the elements of the paths (i.e., segments) as well as an optional Weight sub-TLV.

The Segment List sub-TLV may exceed 255 bytes in length due to a large number of segments. A 2-octet length is thus required. According to section 2 of [RFC9012], the sub-TLV type defines the size of the length field. Therefore, for the Segment List sub-TLV, a code point of 128 or higher is used.

The Segment List sub-TLV is optional and MAY appear multiple times in the SR Policy encoding. The ordering of Segment List sub-TLVs does not matter since each sub-TLV encodes a Segment List.

The Segment List sub-TLV contains zero or more Segment sub-TLVs and MAY contain a Weight sub-TLV.

The Segment List sub-TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |             Length            |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                          sub-TLVs                           //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Segment List sub-TLV

where:

* Type: 128.

* Length: the total length (not including the Type and Length fields) of the sub-TLVs encoded within the Segment List sub-TLV in terms of octets.

* RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

* sub-TLVs currently defined:

  - An optional single Weight sub-TLV.

  - Zero or more Segment sub-TLVs.

Validation of an explicit path encoded by the Segment List sub-TLV is beyond the scope of BGP and performed by the SRPM as described in section 5 of [RFC9256].

2.4.4.1. Weight Sub-TLV

The Weight sub-TLV specifies the weight associated with a given segment list. The contents of this sub-TLV are used only by the SRPM as described in section 2.11 of [RFC9256].

The Weight sub-TLV is optional and it MUST NOT appear more than once inside the Segment List sub-TLV.

The Weight sub-TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Length      |     Flags     |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Weight                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: Weight sub-TLV

where:

* Type: 9.

* Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value MUST be 6.

* Flags: 1 octet of flags. No flags are defined in this document. The Flags field MUST be set to zero on transmission and MUST be ignored on receipt.

* RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

* Weight: 4 octets value indicating the weight associated with a segment list as described in section 2.11 of [RFC9256].

2.4.4.2. Segment Sub-TLVs

A Segment sub-TLV describes a single segment in a segment list (i.e., a single element of the explicit path). One or more Segment sub-TLVs constitute an explicit path of the SR Policy candidate path. The contents of these sub-TLVs are used only by the SRPM as described in section 4 in [RFC9256].

The Segment sub-TLVs are optional and MAY appear multiple times in the Segment List sub-TLV.

Section 4 of [RFC9256] defines several Segment Types:

Type A: SR-MPLS Label
Type B: SRv6 SID
Type C: IPv4 Prefix with optional SR Algorithm
Type D: IPv6 Global Prefix with optional SR Algorithm for SR-MPLS
Type E: IPv4 Prefix with Local Interface ID
Type F: IPv4 Addresses for link endpoints as Local, Remote pair
Type G: IPv6 Prefix and Interface ID for link endpoints as Local, Remote pair for SR-MPLS
Type H: IPv6 Addresses for link endpoints as Local, Remote pair for SR-MPLS
Type I: IPv6 Global Prefix with optional SR Algorithm for SRv6
Type J: IPv6 Prefix and Interface ID for link endpoints as Local, Remote pair for SRv6
Type K: IPv6 Addresses for link endpoints as Local, Remote pair for SRv6

The following sub-sections specify the sub-TLVs used for Segment Types A and B. The other segment types are specified in [I-D.ietf-idr-bgp-sr-segtypes-ext].

2.4.4.2.1. Segment Type A

The Type A Segment Sub-TLV encodes a single SR-MPLS SID. The format is as follows and is used to encode MPLS Label fields as specified in [RFC3032] [RFC5462].:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Length      |     Flags     |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Label                        | TC  |S|       TTL     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Type A Segment sub-TLV

where:

* Type: 1.

* Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value MUST be 6.

* Flags: 1 octet of flags as defined in Section 2.4.4.2.3.

* RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

* Label: 20 bits of label value.

* TC: 3 bits of traffic class.

* S: 1 bit of bottom-of-stack.

* TTL: 1 octet of TTL.

The following applies to the Type-1 Segment sub-TLV:

* The S bit MUST be zero upon transmission and MUST be ignored upon reception.

* If the originator wants the receiver to choose the TC value, it sets the TC field to zero.

* If the originator wants the receiver to choose the TTL value, it sets the TTL field to 255.

* If the originator wants to recommend a value for these fields, it puts those values in the TC and/or TTL fields.

* The receiver MAY override the originator's values for these fields. This would be determined by local policy at the receiver. One possible policy would be to override the fields only if the fields have the default values specified above.

2.4.4.2.2. Segment Type B

The Type B Segment Sub-TLV encodes a single SRv6 SID. The format is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Length      |     Flags     |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                      SRv6 SID (16 octets)                   //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//     SRv6 Endpoint Behavior and SID Structure (optional)     //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: Type B Segment sub-TLV

where:

* Type: 13.

* Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value MUST be 26 when the SRv6 Endpoint Behavior and SID Structure is present else it MUST be 18.

* Flags: 1 octet of flags as defined in Section 2.4.4.2.3.

* RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

* SRv6 SID: 16 octets of IPv6 address.

* SRv6 Endpoint Behavior and SID Structure: Optional, as defined in Section 2.4.4.2.4.

The TLV 2 defined for the advertisement of Segment Type B in the earlier versions of this document has been deprecated to avoid backward compatibility issues.

2.4.4.2.3. Segment Flags

The Segment Types sub-TLVs described above may contain the following flags in the "Flags" field defined in Section 6.8:

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|V|   |B|       |
+-+-+-+-+-+-+-+-+
```

Figure 22: Segment Flags

where:

V-Flag: This flag, when set, is used by SRPM for "SID verification" as described in Section 5.1 of [RFC9256].

B-Flag: This flag, when set, indicates the presence of the SRv6 Endpoint Behavior and SID Structure encoding specified in Section 2.4.4.2.4.

The unassigned bits in the Flag octet MUST be set to zero upon transmission and MUST be ignored upon receipt.

The following applies to the Segment Flags:

* V-Flag applies to all Segment Types.

* B-Flag applies to Segment Type B. If B-Flag appears with Segment Type A it MUST be ignored.

2.4.4.2.4. SRv6 SID Endpoint Behavior and Structure

The Segment Types sub-TLVs described above MAY contain the SRv6 Endpoint Behavior and SID Structure [RFC8986] encoding as described below:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Endpoint Behavior       |            Reserved           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    LB Length  |  LN Length    | Fun. Length   |  Arg. Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 23: SRv6 SID Endpoint Behavior and Structure

where:

Endpoint Behavior: 2 octets. It carries the SRv6 Endpoint Behavior code point for this SRv6 SID as defined in section 9.2 of [RFC8986]. When set with the value 0xFFFF (i.e., Opaque), the choice of SRv6 Endpoint Behavior is left to the headend.

Reserved: 2 octets of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

Locator Block Length: 1 octet. SRv6 SID Locator Block length in bits.

Locator Node Length: 1 octet. SRv6 SID Locator Node length in bits.

Function Length: 1 octet. SRv6 SID Function length in bits.

Argument Length: 1 octet. SRv6 SID Arguments length in bits.

The total of the locator block, locator node, function, and argument lengths MUST be less than or equal to 128.

2.4.5. Explicit NULL Label Policy Sub-TLV

To steer an unlabeled IP packet into an SR policy, it is necessary to create a label stack for that packet, and push one or more labels onto that stack.

The Explicit NULL Label Policy (ENLP) sub-TLV is used to indicate whether an Explicit NULL Label [RFC3032] must be pushed on an unlabeled IP packet before any other labels.

If an ENLP Sub-TLV is not present, the decision of whether to push an Explicit NULL label on a given packet is a matter of local configuration.

The ENLP sub-TLV is optional and it MUST NOT appear more than once in the SR Policy encoding.

The contents of this sub-TLV are used by the SRPM as described in section 4.1 of [RFC9256].

```
0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Length      |     Flags     |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     ENLP      |
+-+-+-+-+-+-+-+-+
```

Figure 24: ELNP sub-TLV

Where:

Type: 14.

Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value MUST be 3.

Flags: 1 octet of flags. No flags are defined in this document. The Flags field MUST be set to zero on transmission and MUST be ignored on receipt.

RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

ENLP (Explicit NULL Label Policy): Indicates whether Explicit NULL labels are to be pushed on unlabeled IP packets that are being steered into a given SR policy. This field has one of the following values:

- 0: Reserved.

- 1: Push an IPv4 Explicit NULL label on an unlabeled IPv4 packet, but do not push an IPv6 Explicit NULL label on an unlabeled IPv6 packet.

- 2: Push an IPv6 Explicit NULL label on an unlabeled IPv6 packet, but do not push an IPv4 Explicit NULL label on an unlabeled IPv4 packet.

- 3: Push an IPv4 Explicit NULL label on an unlabeled IPv4 packet, and push an IPv6 Explicit NULL label on an unlabeled IPv6 packet.

- 4: Do not push an Explicit NULL label.

- 5 - 255: Reserved.

The ENLP reserved values may be used for future extensions and implementations SHOULD ignore the ENLP Sub-TLV with these values. The behavior signaled in this Sub-TLV MAY be overridden by local configuration. The section 4.1 of [RFC9256] describes the behavior on the headend for the handling of the explicit null label.

2.4.6. Policy Priority Sub-TLV

An operator MAY set the Policy Priority sub-TLV to indicate the order in which the SR policies are re-computed upon topological change. The contents of this sub-TLV are used by the SRPM as described in section 2.12 of [RFC9256].

The Priority sub-TLV is optional and it MUST NOT appear more than once in the SR Policy encoding.

The Priority sub-TLV has following format:

```
 0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Type      |   Length      |  Priority     |   RESERVED    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 25: Priority sub-TLV

Where:

Type: 15

Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets.The value MUST be 2.

Priority: a 1-octet value.

RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

2.4.7. Policy Candidate Path Name Sub-TLV

An operator MAY set the Policy Candidate Path Name sub-TLV to attach a symbolic name to the SR Policy candidate path.

Usage of Policy Candidate Path Name sub-TLV is described in section 2.6 of [RFC9256].

The Policy Candidate Path Name sub-TLV may exceed 255 bytes in length due to a long name. A 2-octet length is thus required. According to section 2 of [RFC9012], the sub-TLV type defines the size of the length field. Therefore, for the Policy Candidate Path Name sub-TLV a code point of 128 or higher is used.

It is RECOMMENDED that the size of the symbolic name for the candidate path is limited to 255 bytes. Implementations MAY choose to truncate long names to 255 bytes when signaling via BGP.

The Policy Candidate Path Name sub-TLV is optional and it MUST NOT appear more than once in the SR Policy encoding.

The Policy Candidate Path Name sub-TLV has following format:

```
0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Length                      |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//              Policy Candidate Path Name                     //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 26: Policy Candidate Path Name sub-TLV

Where:

Type: 129.

Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value is variable.

RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

Policy Candidate Path Name: Symbolic name for the SR Policy candidate path without a NULL terminator as specified in section 2.6 of [RFC9256].

2.4.8. Policy Name Sub-TLV

An operator MAY set the Policy Name sub-TLV to associate a symbolic name with the SR Policy for which the candidate path is being advertised via the SR Policy NLRI.

Usage of Policy Name sub-TLV is described in section 2.1 of [RFC9256].

The Policy Name sub-TLV may exceed 255 bytes in length due to a long policy name. A 2-octet length is thus required. According to section 2 of [RFC9012], the sub-TLV type defines the size of the length field. Therefore, for the Policy Name sub-TLV a code point of 128 or higher is used.

It is RECOMMENDED that the size of the symbolic name for the SR Policy is limited to 255 bytes. Implementations MAY choose to truncate long names to 255 bytes when signaling via BGP.

The Policy Name sub-TLV is optional and it MUST NOT appear more than once in the SR Policy encoding.

The Policy Name sub-TLV has following format:

```
0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Length                      |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                        Policy Name                          //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 27: Policy Name sub-TLV

Where:

Type: 130

Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value is variable.

RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

Policy Name: Symbolic name for the policy. It SHOULD be a string of printable ASCII characters, without a NULL terminator.

3. Color Extended Community

The Color Extended Community [RFC9012] is used to steer traffic corresponding to BGP routes into an SR Policy with matching color value. The Color Extended Community MAY be carried in any BGP UPDATE message whose AFI/SAFI is 1/1 (IPv4 Unicast), 2/1 (IPv6 Unicast), 1/4 (IPv4 Labeled Unicast), 2/4 (IPv6 Labeled Unicast), 1/128 (VPN-IPv4 Labeled Unicast), 2/128 (VPN-IPv6 Labeled Unicast), or 25/70 (Ethernet VPN, usually known as EVPN). Use of the Color Extended Community in BGP UPDATE messages of other AFI/SAFIs is outside the scope of this document.

Two bits from the Flags field of the Color Extended Community are used as follows to support the requirements of Color-Only steering as specified in Section 8.8 of [RFC9256]:

```
                     1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|C O|        Unassigned         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 28: Color Extended Community Flags

The CO bits together form the Color-Only Type field which indicates the various matching criteria between BGP NH and SR Policy endpoint in addition to the matching of the color value. Following types are defined:

* Type 0: Specific Endpoint Match: Request match for the endpoint that is the BGP NH

* Type 1: Specific or Null Endpoint Match: Request match for either the endpoint that is the BGP NH or a null endpoint (e.g., like a default gateway)

* Type 2: Specific, Null, or Any Endpoint Match: Request match for either the endpoint that is the BGP NH or with a null or any endpoint

* Type 3: reserved for future use and SHOULD NOT be used. Upon reception, an implementation MUST treat it like Type 0.

The details of the SR Policy steering mechanisms based on these Color-Only types are specified in section 8.8 of [RFC9256].

One or more Color Extended Communities MAY be associated with a BGP route update. Sections 8.4.1, 8.5.1, and 8.8.2 of [RFC9256] specify the steering behaviors over SR Policies when multiple Color Extended Communities are associated with a BGP route.

4. SR Policy Operations

As mentioned in Section 1, BGP is not the actual consumer of an SR Policy NLRI. BGP is in charge of the origination and propagation of the SR Policy NLRI but its installation and use are outside the scope of BGP. The details of SR Policy installation and use are specified in [RFC9256].

4.1. Advertisement of SR Policies

Typically, but not limited to, an SR Policy is computed by a controller or a path computation engine (PCE) and originated by a BGP speaker on its behalf.

Multiple SR Policy NLRIs may be present with the same <color, endpoint> tuple but with different content when these SR policies are intended for different headends.

The distinguisher of each SR Policy NLRI prevents undesired BGP route selection among these SR Policy NLRIs and allows their propagation across route reflectors [RFC4456].

Moreover, one or more route targets SHOULD be attached to the advertisement, where each route target identifies one or more intended headends for the advertised SR Policy update.

If no route target is attached to the SR Policy NLRI, then it is assumed that the originator sends the SR Policy update directly (e.g., through a BGP session) to the intended receiver. In such a case, the NO_ADVERTISE community [RFC1997] MUST be attached to the SR Policy update (see further details in Section 4.2.3).

4.2. Reception of an SR Policy NLRI

On reception of an SR Policy NLRI, a BGP speaker first determines if it is valid as described in Section 4.2.1 and then performs the decision process for selection of the best route (Section 9.1 of [RFC4271]). The key difference from the base BGP decision process is that BGP does not download the selected best routes of SR Policy SAFI into the forwarding and instead considers them "usable" for passing on to the SRPM for further processing as described in Section 4.2.2. The selected best route is propagated (Section 9.1.3 of [RFC4271]) as described in Section 4.2.3 irrespective of its "usability" by the local router.

4.2.1. Validation of an SR Policy NLRI

When a BGP speaker receives an SR Policy NLRI from a neighbor it MUST first perform validation based on the following rules in addition to the validation described in Section 5:

* The SR Policy NLRI MUST include a distinguisher, color, and endpoint field which implies that the length of the NLRI MUST be either 12 or 24 octets (depending on the address family of the endpoint).

* The SR Policy update MUST have either the NO_ADVERTISE community or at least one route target extended community in IPv4-address format or both. If a router supporting this specification receives an SR Policy update with no route target extended communities and no NO_ADVERTISE community, the update MUST be considered as malformed.

* The Tunnel Encapsulation Attribute MUST be attached to the BGP Update and MUST have a Tunnel Type TLV set to SR Policy (codepoint is 15).

A router that receives an SR Policy update that is not valid according to these criteria MUST treat the update as malformed and the SR Policy candidate path MUST NOT be passed to the SRPM.

4.2.2. Eligibility for Local Use of an SR Policy NLRI

An SR Policy NLRI update without any route target extended community but having the NO_ADVERTISE community is considered usable.

If one or more route targets are present, then at least one route target MUST match the BGP Identifier of the receiver for the update to be considered usable. The BGP Identifier is defined in [RFC4271] as a 4-octet IPv4 address. Therefore, the route target extended community MUST be of the same format.

If one or more route targets are present and none matches the local BGP Identifier, then, while the SR Policy NLRI is valid, it is not usable on the receiver node.

When the SR Policy tunnel type includes any sub-TLV that is unrecognized or unsupported, the update SHOULD NOT be considered usable. An implementation MAY provide an option for ignoring unsupported sub-TLVs.

Once BGP on the receiving node has determined that the SR Policy NLRI is usable, it passes the SR Policy candidate path to the SRPM. Note that, along with the candidate path details, BGP also passes the originator information for breaking ties in the candidate path selection process as described in section 2.4 of [RFC9256].

When an update for an SR Policy NLRI results in its becoming unusable, BGP MUST delete its corresponding SR Policy candidate path from the SRPM.

The SRPM applies the rules defined in section 2 of [RFC9256] to determine whether the SR Policy candidate path is valid and to select the best candidate path among the valid ones for a given SR Policy.

4.2.3. Propagation of an SR Policy

SR Policy NLRIs that have the NO_ADVERTISE community attached to them MUST NOT be propagated.

By default, a BGP node receiving an SR Policy NLRI MUST NOT propagate it to any EBGP neighbor. An implementation MAY provide an explicit configuration to override this and enable the propagation of valid SR Policy NLRIs to specific EBGP neighbors where the SR domain comprises multiple-ASes within a single service provider domain (see Section 7 for details).

A BGP node advertises a received SR Policy NLRI to its IBGP neighbors according to normal IBGP propagation rules.

By default, a BGP node receiving an SR Policy NLRI SHOULD NOT remove route target extended community before propagation. An implementation MAY provide support for configuration to filter and/or remove route target extended community before propagation.

A BGP node MUST NOT alter the SR Policy information carried in the Tunnel Encapsulation Attribute during propagation.

5. Error Handling and Fault Management

This section describes the error handling actions, as described in [RFC7606], that are to be performed for the handling of the BGP update messages for BGP SR Policy SAFI.

A BGP Speaker MUST perform the following syntactic validation of the SR Policy NLRI to determine if it is malformed. This includes the validation of the length of each NLRI and the total length of the MP_REACH_NLRI and MP_UNREACH_NLRI attributes. It also includes the validation of the consistency of the NLRI length with the AFI and the endpoint address as specified in Section 2.1.

When the error determined allows for the router to skip the malformed NLRI(s) and continue the processing of the rest of the update message, then it MUST handle such malformed NLRIs as 'Treat-as-withdraw'. In other cases, where the error in the NLRI encoding results in the inability to process the BGP update message (e.g. length related encoding errors), then the router SHOULD handle such malformed NLRIs as 'AFI/SAFI disable' when other AFI/SAFI besides SR Policy are being advertised over the same session. Alternately, the router MUST perform 'session reset' when the session is only being used for SR Policy or when it 'AFI/SAFI disable' action is not possible.

The validation of the TLVs/sub-TLVs introduced in this document and defined in their respective sub-sections of Section 2.4 MUST be performed to determine if they are malformed or invalid. The validation of the Tunnel Encapsulation Attribute itself and the other TLVs/sub-TLVs specified in [RFC9012] MUST be done as described in that document. In case of any error detected, either at the attribute or its TLV/sub-TLV level, the "treat-as-withdraw" strategy MUST be applied. This is because an SR Policy update without a valid Tunnel Encapsulation Attribute (comprising of all valid TLVs/sub-TLVs) is not usable.

An SR Policy update that is determined to be not valid, and therefore malformed, based on rules described in Section 4.2.1 MUST be handled by the "treat-as-withdraw" strategy.

The validation of the individual fields of the TLVs/sub-TLVs defined in Section 2.4 are beyond the scope of BGP as they are handled by the SRPM as described in the individual TLV/sub-TLV sub-sections. A BGP implementation MUST NOT perform semantic verification of such fields nor consider the SR Policy update to be invalid or not usable based on such validation.

An implementation SHOULD log any errors found during the above validation for further analysis.

6. IANA Considerations

This document uses code point allocations from the following existing registries:

* Subsequent Address Family Identifiers (SAFI) Parameters registry

* BGP Tunnel Encapsulation Attribute Tunnel Types registry under the BGP Tunnel Encapsulation registry

* BGP Tunnel Encapsulation Attribute sub-TLVs registry under the BGP Tunnel Encapsulation registry

* Color Extended Community Flags registry under the BGP Tunnel Encapsulation registry

This document also requests the creation of the following new registries:

* SR Policy Segment List Sub-TLVs under the BGP Tunnel Encapsulation registry

* SR Policy Binding SID Flags under the BGP Tunnel Encapsulation registry

* SR Policy SRv6 Binding SID Flags under the BGP Tunnel Encapsulation registry

* SR Policy Segment Flags under the BGP Tunnel Encapsulation registry

* Color Extended Community Color-Only Types registry under the BGP Tunnel Encapsulation registry

6.1. Existing Registry: Subsequent Address Family Identifiers (SAFI) Parameters

This document introduces a SAFI in the registry "Subsequent Address Family Identifiers (SAFI) Parameters" that has been assigned a code point by IANA. The entry needs to be updated as follows:

Code Point	Description	Reference
73	SR Policy SAFI	This document

Table 1: BGP SAFI Code Point

6.2. Existing Registry: BGP Tunnel Encapsulation Attribute Tunnel Types

This document introduces a Tunnel-Type in the registry "BGP Tunnel Encapsulation Attribute Tunnel Types" that has been assigned a codepoint by IANA. The entry needs to be updated as follows:

Code Point	Description	Reference
15	SR Policy	This document

Table 2: Tunnel Type Code Point

6.3. Existing Registry: BGP Tunnel Encapsulation Attribute sub-TLVs

This document defines sub-TLVs in the registry "BGP Tunnel Encapsulation Attribute sub-TLVs" that have been assigned code points by IANA as follows via the early allocation process which needs to be made permanent:

Code Point	Description	Reference
12	Preference sub-TLV	This document
13	Binding SID sub-TLV	This document
14	ENLP sub-TLV	This document
15	Priority sub-TLV	This document
20	SRv6 Binding SID sub-TLV	This document
128	Segment List sub-TLV	This document
129	Policy Candidate Path Name sub-TLV	This document
130	Policy Name sub-TLV	This document

Table 3: BGP Tunnel Encapsulation Attribute Code Points

6.4. Existing Registry: Color Extended Community Flags

This document defines the use of 2 bits in the registry called "Color Extended Community Flags" under the "BGP Tunnel Encapsulation" registry that have been assigned by IANA via the early allocation process to form the Color-Only Types field which needs to be made permanent:

Bit Position	Description	Reference
0-1	Color-only Types Field	This document

Table 4: Color Extended Community Flag Bits

6.5. New Registry: SR Policy Segment List Sub-TLVs

This document requests the creation of a new registry called "SR Policy Segment List Sub-TLVs" under the "BGP Tunnel Encapsulation" registry. The allocation policy of this registry is "Standards Action" according to [RFC8126].

Following initial Sub-TLV codepoints are assigned by this document:

Value	Description	Reference
0	Reserved	This document
1	Segment Type A sub-TLV	This document
2	Deprecated	This document
3-8	Unassigned	
9	Weight sub-TLV	This document
10	Deprecated	This document
11	Deprecated	This document
12	Deprecated	This document
13	Segment Type B sub-TLV	This document
14-255	Unassigned	

Table 5: SR Policy Segment List Code Points

6.6. New Registry: SR Policy Binding SID Flags

This document requests the creation of a new registry called "SR Policy Binding SID Flags" under the "BGP Tunnel Encapsulation" registry. The allocation policy of this registry is "Standards Action" according to [RFC8126].

The following flags are defined:

Bit	Description	Reference
0	Specified-BSID-Only Flag (S-Flag)	This document
1	Drop Upon Invalid Flag (I-Flag)	This document
2-7	Unassigned	

Table 6: SR Policy Binding SID Flags

6.7. New Registry: SR Policy SRv6 Binding SID Flags

This document requests the creation of a new registry called "SR Policy SRv6 Binding SID Flags" under the "BGP Tunnel Encapsulation" registry. The allocation policy of this registry is "Standards Action" according to [RFC8126].

The following flags are defined:

Bit	Description	Reference
0	Specified-BSID-Only Flag (S-Flag)	This document
1	Drop Upon Invalid Flag (I-Flag)	This document
2	SRv6 Endpoint Behavior & SID Structure Flag (B-Flag)	This document
3-7	Unassigned	

Table 7: SR Policy SRv6 Binding SID Flags

6.8. New Registry: SR Policy Segment Flags

This document requests the creation of a new registry called "SR Policy Segment Flags" under the "BGP Tunnel Encapsulation" registry. The allocation policy of this registry is "Standards Action" according to [RFC8126].

The following flags are defined:

Bit	Description	Reference
0	Segment Verification Flag (V-Flag)	This document
1-2	Unassigned	
3	SRv6 Endpoint Behavior & SID Structure Flag (B-Flag)	This document
4-7	Unassigned	

Table 8: SR Policy Segment Flags

6.9. New Registry: Color Extended Community Color-Only Types

This document requests the creation of a new registry called "Color Extended Community Color-Only Types" under the "BGP Tunnel Encapsulation" registry for assignment of codepoints (values 0 through 3) in the Color-Only Type field of the Color Extended Community Flags field. The allocation policy of this registry is "Standards Action" according to [RFC8126].

The following types are defined:

Type	Description	Reference
0	Specific Endpoint Match	This document
1	Specific or Null Endpoint Match	This document
2	Specific, Null, or Any Endpoint Match	This document
3	Unassigned	This document

Table 9: Color Extended Community Color-Only Types

7. Security Considerations

The security mechanisms of the base BGP security model apply to the extensions described in this document as well. See the Security Considerations section of [RFC4271] for a discussion of BGP security. Also, refer to [RFC4272] and [RFC6952] for analysis of security issues for BGP.

The BGP SR Policy extensions specified in this document enable traffic engineering and service programming use-cases within an SR domain as described in [RFC9256]. SR operates within a trusted SR domain [RFC8402] and its security considerations also apply to BGP sessions when carrying SR Policy information. The SR Policies distributed by BGP are expected to be used entirely within this trusted SR domain which comprises a single AS or multiple ASes/ domains within a single provider network. Therefore, precaution is necessary to ensure that the SR Policy information advertised via BGP sessions is limited to nodes in a secure manner within this trusted SR domain. BGP peering sessions for address-families other than SR Policy SAFI may be set up to routers outside the SR domain. The isolation of BGP SR Policy SAFI peering sessions may be used to ensure that the SR Policy information is not advertised by accident or error to an EBGP peering session outside the SR domain.

Additionally, it may be considered that the export of SR Policy information, as described in this document, constitutes a risk to confidentiality of mission-critical or commercially sensitive information about the network (more specifically endpoint/node

addresses, SR SIDs, and the SR Policies deployed). BGP peerings are not automatic and require configuration; thus, it is the responsibility of the network operator to ensure that only trusted nodes (that include both routers and controller applications) within the SR domain are configured to receive such information.

8. Manageability Considerations

The specification of BGP models is an ongoing work based on [I-D.ietf-idr-bgp-model] and its future extensions are expected to cover the SR Policy SAFI. Existing BGP operational procedures also apply to the SAFI specified in this document. The management, operations, and monitoring of BGP speakers and the SR Policy SAFI sessions between them are not very different from other BGP sessions and can be managed using the same data models.

The YANG model for the operation and management of SR Policies [I-D.ietf-spring-sr-policy-yang] reports the SR Policies provisioned via BGP SR Policy SAFI along with their operational states.

9. Acknowledgments

The authors of this document would like to thank Shyam Sethuram, John Scudder, Przemyslaw Krol, Alex Bogdanov, Nandan Saha, Bruno Decraene, Gurusiddesh Nidasesi, Kausik Majumdar, Zafar Ali, Swadesh Agarwal, Jakob Heitz, Viral Patel, Peng Shaofu, Cheng Li, Martin Vigoureux, John Scudder, Vincent Roca, Brian Haberman, Mohamed Boucadair and Shunwan Zhuang for their comments and review of this document. The authors would like to thank Sue Hares for her detailed shepherd review that helped in improving the document. The authors would also like to thank Andrew Alston for his detailed AD review and comments.

10. Contributors

Eric Rosen
Juniper Networks
US

Email: erosen@juniper.net

Arjun Sreekantiah
Cisco Systems
US

Email: asreekan@cisco.com

Acee Lindem
Cisco Systems
US

Email: acee@cisco.com

Siva Sivabalan
Cisco Systems
US

Email: msiva@cisco.com

Imtiyaz Mohammad
Arista Networks
India

Email: imtiyaz@arista.com

Gaurav Dawra
Cisco Systems
US

Email: gdawra.ietf@gmail.com

Peng Shaofu
ZTE Corporation
China

Email: peng.shaofu@zte.com.cn

Steven Lin
Calix
USA

Email: steven.lin@calix.com

11. References

11.1. Normative References

[RFC1997] Chandra, R., Traina, P., and T. Li, "BGP Communities Attribute", RFC 1997, DOI 10.17487/RFC1997, August 1996, <https://www.rfc-editor.org/info/rfc1997>.

[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>.

[RFC2545] Marques, P. and F. Dupont, "Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing", RFC 2545, DOI 10.17487/RFC2545, March 1999, <https://www.rfc-editor.org/info/rfc2545>.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <https://www.rfc-editor.org/info/rfc3032>.

[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>.

[RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, DOI 10.17487/RFC4360, February 2006, <https://www.rfc-editor.org/info/rfc4360>.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <https://www.rfc-editor.org/info/rfc4760>.

[RFC5462] Andersson, L. and R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, DOI 10.17487/RFC5462, February 2009, <https://www.rfc-editor.org/info/rfc5462>.

[RFC7606] Chen, E., Ed., Scudder, J., Ed., Mohapatra, P., and K. Patel, "Revised Error Handling for BGP UPDATE Messages", RFC 7606, DOI 10.17487/RFC7606, August 2015, <https://www.rfc-editor.org/info/rfc7606>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC8660] Bashandy, A., Ed., Filsfils, C., Ed., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with the MPLS Data Plane", RFC 8660, DOI 10.17487/RFC8660, December 2019, <https://www.rfc-editor.org/info/rfc8660>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

[RFC9012] Patel, K., Van de Velde, G., Sangli, S., and J. Scudder, "The BGP Tunnel Encapsulation Attribute", RFC 9012, DOI 10.17487/RFC9012, April 2021, <https://www.rfc-editor.org/info/rfc9012>.

[RFC9256] Filsfils, C., Talaulikar, K., Ed., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", RFC 9256, DOI 10.17487/RFC9256, July 2022, <https://www.rfc-editor.org/info/rfc9256>.

11.2. Informational References

[I-D.ietf-idr-bgp-model]
Jethanandani, M., Patel, K., Hares, S., and J. Haas, "YANG Model for Border Gateway Protocol (BGP-4)", Work in Progress, Internet-Draft, draft-ietf-idr-bgp-model-17, 5 July 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-bgp-model-17>.

[I-D.ietf-idr-bgp-sr-segtypes-ext]
Talaulikar, K., Filsfils, C., Previdi, S., Mattes, P., and D. Jain, "Segment Routing Segment Types Extensions for BGP SR Policy", Work in Progress, Internet-Draft, draft-ietf-idr-bgp-sr-segtypes-ext-01, 26 September 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-bgp-sr-segtypes-ext-01>.

[I-D.ietf-spring-sr-policy-yang]
Raza, S., Sawaya, R., Shunwan, Z., Voyer, D., Durrani, M., Matsushima, S., and V. P. Beeram, "YANG Data Model for Segment Routing Policy", Work in Progress, Internet-Draft,

draft-ietf-spring-sr-policy-yang-02, 23 September 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-spring-sr-policy-yang-02>.

[RFC4272] Murphy, S., "BGP Security Vulnerabilities Analysis", RFC 4272, DOI 10.17487/RFC4272, January 2006, <https://www.rfc-editor.org/info/rfc4272>.

[RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, DOI 10.17487/RFC4456, April 2006, <https://www.rfc-editor.org/info/rfc4456>.

[RFC6952] Jethanandani, M., Patel, K., and L. Zheng, "Analysis of BGP, LDP, PCEP, and MSDP Issues According to the Keying and Authentication for Routing Protocols (KARP) Design Guide", RFC 6952, DOI 10.17487/RFC6952, May 2013, <https://www.rfc-editor.org/info/rfc6952>.

Authors' Addresses

Stefano Previdi
Huawei Technologies
Italy
Email: stefano@previdi.net

Clarence Filsfils
Cisco Systems
Brussels
Belgium
Email: cfilsfil@cisco.com

Ketan Talaulikar (editor)
Cisco Systems
India
Email: ketant.ietf@gmail.com

Paul Mattes
Microsoft
One Microsoft Way
Redmond, WA 98052
United States of America
Email: pamattes@microsoft.com

Dhanendra Jain
Google
Email: dhanendra.ietf@gmail.com

Inter-Domain Routing
Internet-Draft
Intended status: Standards Track
Expires: 29 July 2023

S. Previdi
K. Talaulikar, Ed.
Cisco Systems
J. Dong, Ed.
M. Chen
Huawei Technologies
H. Gredler
RtBrick Inc.
J. Tantsura
Microsoft

25 January 2023

Distribution of Traffic Engineering (TE) Policies and State using BGP-LS
draft-ietf-idr-te-lsp-distribution-19

Abstract

This document describes a mechanism to collect the Traffic Engineering Policy information that is locally available in a node and advertise it into BGP Link State (BGP-LS) updates. Such information can be used by external components for path computation, re-optimization, service placement, network visualization, etc.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In many network environments, traffic engineering (TE) policies are instantiated into various forms:

* MPLS Traffic Engineering Label Switched Paths (TE-LSPs).

* Segment Routing (SR) Policies as defined in [RFC9256]

* Local MPLS cross-connect configuration

All this information can be grouped into the same term: TE Policies. In the rest of this document we refer to TE Policies as the set of information related to the various instantiation of policies: MPLS TE LSPs, SR Policies, etc.

SR Policy architecture details are specified in [RFC9256]. An SR Policy comprises one or more candidate paths (CP) of which at a given time one and only one may be active (i.e., installed in forwarding and usable for steering of traffic). Each CP in turn may have one or more SID-List of which one or more may be active; when multiple are active then traffic is load balanced over them. This document covers the advertisement of state information at the individual SR Policy CP level.

TE Policies are generally instantiated at the head-end and are based on either local configuration or controller-based programming of the node using various APIs and protocols, e.g., PCEP or BGP.

In many network environments, the configuration, and state of each TE Policy that is available in the network is required by a controller which allows the network operator to optimize several functions and operations through the use of a controller aware of both topology and state information.

One example of a controller is the stateful Path Computation Element (PCE) [RFC8231], which could provide benefits in path optimization. While some extensions are proposed in the Path Computation Element Communication Protocol (PCEP) for the Path Computation Clients (PCCs) to report the LSP states to the PCE, this mechanism may not be applicable in a management-based PCE architecture as specified in section 5.5 of [RFC4655]. As illustrated in the figure below, the PCC is not an LSR in the routing domain, thus the head-end nodes of the TE-LSPs may not implement the PCEP protocol. In this case, a general mechanism to collect the TE-LSP states from the ingress LERs is needed. This document proposes a TE Policy state collection mechanism complementary to the mechanism defined in [RFC8231].

```
                                  -----------
                                 |   -----   |
            Service              |  | TED |<-+----------->
            Request              |   -----   |  TED synchronization
               |                 |     |     |  mechanism (e.g.,
               v                 |     |     |  routing protocol)
         ------------- Request/  |     v     |
        |             | Response|   -----   |
        |     NMS     |<--------+> | PCE |  |
        |             |         |   -----   |
         -------------           -----------
       Service |
       Request |
               v
          ----------  Signaling   ----------
         | Head-End | Protocol   | Adjacent |
         |  Node    |<---------->|   Node   |
          ----------              ----------
```

Figure 1. Management-Based PCE Usage

In networks with composite PCE nodes as specified in section 5.1 of [RFC4655], PCE is implemented on several routers in the network, and the PCCs in the network can use the mechanism described in [RFC8231] to report the TE Policy information to the PCE nodes. An external component may also need to collect the TE Policy information from all the PCEs in the network to obtain a global view of the LSP state in the network.

In multi-area or multi-AS scenarios, each area or AS can have a child PCE to collect the TE Policies in its domain, in addition, a parent PCE needs to collect TE Policy information from multiple child PCEs to obtain a global view of LSPs inside and across the domains involved.

In another network scenario, a centralized controller is used for service placement. Obtaining the TE Policy state information is quite important for making appropriate service placement decisions with the purpose of both meeting the application's requirements and utilizing network resources efficiently.

The Network Management System (NMS) may need to provide global visibility of the TE Policies in the network as part of the network visualization function.

BGP has been extended to distribute link-state and traffic engineering information to external components [RFC7752]. Using the same protocol to collect Traffic Engineering Policy and state information is desirable for these external components since this avoids introducing multiple protocols for network information collection. This document describes a mechanism to distribute traffic engineering policy information (MPLS, SR, IPv4, and IPv6) to external components using BGP-LS.

1.1. Requirements Language

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.

2. Carrying TE Policy Information in BGP

TE Policy information is advertised in BGP UPDATE messages using the MP_REACH_NLRI and MP_UNREACH_NLRI attributes [RFC4760]. The "Link-State NLRI" defined in [RFC7752] is extended to carry the TE Policy information. BGP speakers that wish to exchange TE Policy information MUST use the BGP Multiprotocol Extensions Capability Code (1) to advertise the corresponding (AFI, SAFI) pair, as specified in [RFC4760]. New TLVs carried in the Link_State Attribute defined in [RFC7752] are also defined to carry the attributes of a TE Policy in the subsequent sections.

The format of "Link-State NLRI" is defined in [RFC7752] as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            NLRI Type          |     Total NLRI Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
//                  Link-State NLRI (variable)                  //
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Additional "NLRI Types" are defined for TE Policy Information as following:

* SR Policy Candidate Path NLRI (value 5)

* MPLS-TE LSP NLRI (value TBD)

* MPLS Local Cross-connect NLRI (value TBD)

The SR Policy Candidate Path NLRI is used to report the state details of individual SR Policy Candidate paths along with their underlying segment lists.

The common format for these NLRI types is defined in Section 3 below.

3. TE Policy NLRI Types

This document defines TE Policy NLRI Types with their common format as shown in the following figure:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|  Protocol-ID  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Identifier                             |
|                        (64 bits)                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//           Node Descriptor TLV (for the Headend)             //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                TE Policy Descriptors (variable)             //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Protocol-ID field specifies the component that owns the TE Policy state in the advertising node. The existing protocol-id value of 5 for Static Configuration applies for some of the NLRI types and the following additional Protocol-IDs are defined for some of the other types in this document:

Protocol-ID	NLRI information source protocol
8	RSVP-TE
9	Segment Routing

* "Identifier" is an 8 octet value as defined in [RFC7752].

* "Local Node Descriptor" (TLV 256) as defined in [RFC7752] that describes the headend node.

* "TE Policy Descriptors" consists of one or more of the TLVs listed as below for use with the respective NLRI type advertisements as specified in Section 4:

Codepoint	Descriptor TLVs
550	Tunnel ID
551	LSP ID
552	IPv4/6 Tunnel Head-end address
553	IPv4/6 Tunnel Tail-end address
554	SR Policy Candidate Path
555	Local MPLS Cross Connect

The Local Node Descriptor TLV MUST include the following Node Descriptor TLVs:

* BGP Router-ID (TLV 516) [RFC9086], which contains a valid BGP Identifier of the local node.

* Autonomous System Number (TLV 512) [RFC7752], which contains the ASN or AS Confederation Identifier (ASN) [RFC5065], if confederations are used, of the local node.

The Local Node Descriptor TLV SHOULD include at least one of the following Node Descriptor TLVs:

* IPv4 Router-ID of Local Node (TLV 1028) [RFC7752], which contains the IPv4 TE Router-ID of the local node when one is provisioned.

* IPv6 Router-ID of Local Node (TLV 1029) [RFC7752], which contains the IPv6 TE Router-ID of the local node when one is provisioned.

The Local Node Descriptor TLV MAY include the following Node Descriptor TLVs:

* BGP Confederation Member (TLV 517) [RFC9086], which contains the ASN of the confederation member (i.e. Member-AS Number), if BGP confederations are used, of the local node.

* Node Descriptors as defined in [RFC7752].

4. TE Policy Descriptors

This section defines the TE Policy Descriptors TLVs which are used to describe the TE Policy being advertised by using the NLRI types defined in Section 3.

4.1. Tunnel Identifier

The Tunnel Identifier TLV contains the Tunnel ID defined in [RFC3209] and is used with the Protocol-ID set to RSVP-TE to advertise the MPLS-TE LSP NLRI Type. It is a mandatory TE policy descriptor TLV for MPLS-TE LSP NLRI type. It has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tunnel ID             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 550

* Length: 2 octets.

* Tunnel ID: 2 octets as defined in [RFC3209].

4.2. LSP Identifier

The LSP Identifier TLV contains the LSP ID defined in [RFC3209] and is used with the Protocol-ID set to RSVP-TE to advertise the MPLS-TE LSP NLRI Type. It is a mandatory TE policy descriptor TLV for MPLS-TE LSP NLRI type. It has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            LSP ID             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 551

* Length: 2 octets.

* LSP ID: 2 octets as defined in [RFC3209].

4.3. IPv4/IPv6 Tunnel Head-End Address

The IPv4/IPv6 Tunnel Head-End Address TLV contains the Tunnel Head-End Address defined in [RFC3209] and is used with the Protocol-ID set to RSVP-TE to advertise the MPLS-TE LSP NLRI Type. It is a mandatory TE policy descriptor TLV for MPLS-TE LSP NLRI type. It has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//        IPv4/IPv6 Tunnel Head-End Address (variable)         //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 552

* Length: 4 or 16 octets.

When the IPv4/IPv6 Tunnel Head-end Address TLV contains an IPv4 address, its length is 4 (octets).

When the IPv4/IPv6 Tunnel Head-end Address TLV contains an IPv6 address, its length is 16 (octets).

4.4. IPv4/IPv6 Tunnel Tail-End Address

The IPv4/IPv6 Tunnel Tail-End Address TLV contains the Tunnel Tail-End Address defined in [RFC3209] and is used with the Protocol-ID set to RSVP-TE to advertise the MPLS-TE LSP NLRI Type. It is a mandatory TE policy descriptor TLV for MPLS-TE LSP NLRI type. It has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//        IPv4/IPv6 Tunnel Tail-End Address (variable)         //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 553

* Length: 4 or 16 octets.

When the IPv4/IPv6 Tunnel Tail-end Address TLV contains an IPv4 address, its length is 4 (octets).

When the IPv4/IPv6 Tunnel Tail-end Address TLV contains an IPv6 address, its length is 16 (octets).

4.5. SR Policy Candidate Path Descriptor

The SR Policy Candidate Path Descriptor TLV identifies a Segment Routing Policy Candidate Path (CP) as defined in [RFC9256]. It is used with the Protocol-ID set to Segment Routing to advertise the SR Policy Candidate Path NLRI Type. It is a mandatory TE policy descriptor TLV for SR Policy Candidate Path NLRI type. The TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Protocol-origin|    Flags      |            RESERVED           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Endpoint (4 or 16 octets)                //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Policy Color (4 octets)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Originator AS Number (4 octets)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Originator Address (4 or 16 octets)             //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Discriminator (4 octets)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 554

* Length: variable (valid values are 24, 36 or 48 octets)

* Protocol-Origin: 1-octet field which identifies the protocol or component which is responsible for the instantiation of this path. Following protocol-origin codepoints are defined in this document.

Code Point	Protocol Origin
1	PCEP
2	BGP SR Policy
3	Configuration (CLI, YANG model via NETCONF, etc.)

* Flags: 1-octet field with following bit positions defined. Other bits MUST be cleared by the originator and MUST be ignored by a receiver.

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|E|O|           |
+-+-+-+-+-+-+-+-+
```

where:

- E-Flag: Indicates the encoding of endpoint as IPv6 address when set and IPv4 address when clear

- O-Flag: Indicates the encoding of originator address as IPv6 address when set and IPv4 address when clear

* Reserved: 2 octets which MUST be set to 0 by the originator and MUST be ignored by a receiver.

* Endpoint: 4 or 16 octets (as indicated by the flags) containing the address of the endpoint of the SR Policy

* Color: 4 octets that indicate the color of the SR Policy

* Originator ASN: 4 octets to carry the 4-byte encoding of the ASN of the originator. Refer to section 2.4 of [RFC9256] for details.

* Originator Address: 4 or 16 octets (as indicated by the flags) to carry the address of the originator. Refer to section 2.4 of [RFC9256] for details.

* Discriminator: 4 octets to carry the discriminator of the path. Refer to section 2.5 of [RFC9256] for details.

4.6. Local MPLS Cross Connect

The Local MPLS Cross Connect TLV identifies a local MPLS state in the form of an incoming label and interface followed by an outgoing label and interface. The outgoing interface may appear multiple times (for multicast states). It is used with Protocol ID set to "Static Configuration" value 5 as defined in [RFC7752]. It is a mandatory TE policy descriptor TLV for MPLS Local Cross-connect NLRI type.

The Local MPLS Cross Connect TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Incoming label (4 octets)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Outgoing label (4 octets)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                          Sub-TLVs (variable)                //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 555

* Length: variable.

* Incoming and Outgoing labels: 4 octets each.

* Sub-TLVs: following Sub-TLVs are defined:

  - Interface Sub-TLV

  - Forwarding Equivalent Class (FEC)

The Local MPLS Cross Connect TLV:

MUST have an incoming label.

MUST have an outgoing label.

MAY contain an Interface Sub-TLV having the I-flag set.

MUST contain at least one Interface Sub-TLV having the I-flag unset.

MAY contain multiple Interface Sub-TLV having the I-flag unset. This is the case of a multicast MPLS cross-connect.

MAY contain an FEC Sub-TLV.

The following sub-TLVs are defined for the Local MPLS Cross Connect TLV:

Codepoint	Descriptor TLV
556	MPLS Cross Connect Interface
557	MPLS Cross Connect FEC

These are defined in the following sub-sections.

4.6.1. MPLS Cross Connect Interface

The MPLS Cross Connect Interface sub-TLV is optional and contains the identifier of the interface (incoming or outgoing) in the form of an IPv4/IPv6 address and/or a local interface identifier.

The MPLS Cross Connect Interface sub-TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+
|     Flags     |
+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Local Interface Identifier (4 octets)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//         Interface Address (4 or 16 octets)                  //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 556

* Length: 9 or 21.

* Flags: 1 octet of flags defined as follows:

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|I|             |
+-+-+-+-+-+-+-+-+
```

where:

  - I-Flag is the Interface flag. When set, the Interface Sub-TLV describes an incoming interface. If the I-flag is not set, then the Interface Sub-TLV describes an outgoing interface.

* Local Interface Identifier: a 4-octet identifier.

* Interface address: a 4-octet IPv4 address or a 16-octet IPv6 address.

4.6.2. MPLS Cross Connect FEC

The MPLS Cross Connect FEC sub-TLV is optional and contains the FEC associated with the incoming label.

The MPLS Cross Connect FEC sub-TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Flags       |  Masklength   |   Prefix (variable)          //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                     Prefix (variable)                       //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 557

* Length: variable.

* Flags: 1 octet of flags defined as follows:

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|4|             |
+-+-+-+-+-+-+-+-+
```

where:

  - 4-Flag is the IPv4 flag. When set, the FEC Sub-TLV describes an IPv4 FEC. If the 4-flag is not set, then the FEC Sub-TLV describes an IPv6 FEC.

* Mask Length: 1 octet of prefix length.

* Prefix: an IPv4 or IPv6 prefix whose mask length is given by the " Mask Length" field padded to an octet boundary.

5. MPLS-TE Policy State TLV

A new TLV called "MPLS-TE Policy State TLV", is used to describe the characteristics of the MPLS-TE LSP NLRI type and it is carried in the optional non-transitive BGP Attribute "LINK_STATE Attribute" defined in [RFC7752]. These MPLS-TE LSP characteristics include the characteristics and attributes of the LSP, its dataplane, explicit path, Quality of Service (QoS) parameters, route information, the protection mechanisms, etc.

The MPLS-TE Policy State TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Object-origin | Address Family|            RESERVED           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//        MPLS-TE Policy State Objects (variable)              //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

Figure 1: MPLS-TE Policy State TLV

* Type: 1200

* Length: the total length of the MPLS-TE Policy State TLV not including the Type and Length fields.

* Object-origin: identifies the component (or protocol) from which the contained object originated. This allows for objects defined in different components to be collected while avoiding the possible codepoint collisions among these components. The following object-origin codepoints are defined in this document.

Code Point	Object Origin
1	RSVP-TE
2	PCEP
3	Local/Static

* Address Family: describes the address family used to set up the MPLS-TE policy. The following address family values are defined in this document:

Code Point	Dataplane
1	MPLS-IPv4
2	MPLS-IPv6

* RESERVED: 16-bit field. SHOULD be set to 0 on transmission and MUST be ignored on receipt.

* TE Policy State Objects: Rather than replicating all these objects in this document, the semantics and encodings of the objects as defined in RSVP-TE and PCEP are reused.

The state information is carried in the "MPLS-TE Policy State Objects" with the following format as described in the sub-sections below.

5.1. RSVP Objects

RSVP-TE objects are encoded in the "MPLS-TE Policy State Objects" field of the MPLS-TE Policy State TLV and consists of MPLS TE LSP objects defined in RSVP-TE [RFC3209] [RFC3473]. Rather than replicating all MPLS TE LSP-related objects in this document, the semantics and encodings of the MPLS TE LSP objects are re-used. These MPLS TE LSP objects are carried in the MPLS-TE Policy State TLV.

When carrying RSVP-TE objects, the "Object-Origin" field is set to "RSVP-TE".

The following RSVP-TE Objects are defined:

* SENDER_TSPEC and FLOW_SPEC [RFC2205]

* SESSION_ATTRIBUTE [RFC3209]

* EXPLICIT_ROUTE Object (ERO) [RFC3209]

* ROUTE_RECORD Object (RRO) [RFC3209]

* FAST_REROUTE Object [RFC4090]

* DETOUR Object [RFC4090]

* EXCLUDE_ROUTE Object (XRO) [RFC4874]

* SECONDARY_EXPLICIT_ROUTE Object (SERO) [RFC4873]

* SECONDARY_RECORD_ROUTE (SRRO) [RFC4873]

* LSP_ATTRIBUTES Object [RFC5420]

* LSP_REQUIRED_ATTRIBUTES Object [RFC5420]

* PROTECTION Object [RFC3473][RFC4872][RFC4873]

* ASSOCIATION Object [RFC4872]

* PRIMARY_PATH_ROUTE Object [RFC4872]

* ADMIN_STATUS Object [RFC3473]

* LABEL_REQUEST Object [RFC3209][RFC3473]

For the MPLS TE LSP Objects listed above, the corresponding sub-objects are also applicable to this mechanism. Note that this list is not exhaustive, other MPLS TE LSP objects which reflect specific characteristics of the MPLS TE LSP can also be carried in the LSP state TLV.

5.2. PCEP Objects

PCEP objects are encoded in the "MPLS-TE Policy State Objects" field of the MPLS-TE Policy State TLV and consist of PCEP objects defined in [RFC5440]. Rather than replicating all MPLS TE LSP-related objects in this document, the semantics, and encodings of the MPLS TE LSP objects are re-used. These MPLS TE LSP objects are carried in the MPLS-TE Policy State TLV.

When carrying PCEP objects, the "Object-Origin" field is set to "PCEP".

The following PCEP Objects are defined:

* METRIC Object [RFC5440]

* BANDWIDTH Object [RFC5440]

For the MPLS TE LSP Objects listed above, the corresponding sub-objects are also applicable to this mechanism. Note that this list is not exhaustive, other MPLS TE LSP objects which reflect specific characteristics of the MPLS TE LSP can also be carried in the TE Policy State TLV.

6. SR Policy State TLVs

This section defines the various TLVs which enable the headend to report the state at the SR Policy CP level. These TLVs (and their sub-TLVs) are carried in the optional non-transitive BGP Attribute "LINK_STATE Attribute" defined in [RFC7752] associated with the SR Policy CP NLRI type.

The detailed procedures for the advertisement are described in Section 7.

6.1. SR Binding SID TLV

The SR Binding SID (BSID) is an optional TLV that is used to report the BSID and its attributes for the SR Policy CP. The TLV MAY also optionally contain the Specified BSID value for reporting as described in section 6.2.3 of [RFC9256]. Only a single instance of this TLV is advertised for a given CP. If multiple instances are present, then the first one is considered valid and the rest are ignored.

The TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           BSID Flags          |            RESERVED           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Binding SID (4 or 16 octets)               //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Specified Binding SID (4 or 16 octets)         //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 1201

* Length: variable (valid values are 12 or 36 octets)

* BSID Flags: 2-octet field that indicates attribute and status of the Binding SID (BSID) associated with this CP. The following bit positions are defined and the semantics are described in detail in [RFC9256]. Other bits MUST be cleared by the originator and MUST be ignored by a receiver.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|D|B|U|L|F|                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- D-Flag: Indicates the dataplane for the BSIDs and if they are 16 octet SRv6 SID when set and are 4 octet SR/MPLS label value when clear.

- B-Flag: Indicates the allocation of the value in the BSID field when set and indicates that BSID is not allocated when clear.

- U-Flag: Indicates the specified BSID value is unavailable when set.

- L-Flag: Indicates the BSID value is from the Segment Routing Local Block (SRLB) of the headend node when set and is from the local dynamic label pool when clear

- F-Flag: Indicates the BSID value is one allocated from dynamic label pool due to fallback (e.g. when specified BSID is unavailable) when set.

* RESERVED: 2 octets. MUST be set to 0 by the originator and MUST be ignored by a receiver.

* Binding SID: It indicates the operational or allocated BSID value based on the status flags.

* Specified BSID: It is used to report the explicitly specified BSID value regardless of whether it is successfully allocated or not. The field is set to value 0 when BSID has not been specified.

The BSID fields above are 4-octet carrying the MPLS Label or 16-octet carrying the SRv6 SID based on the BSID D-flag. When carrying the MPLS Label, as shown in the figure below, the TC, S, and TTL (total of 12 bits) are RESERVED and MUST be set to 0 by the originator and MUST be ignored by a receiver.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Label                        | TC  |S|       TTL     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

In the case of an SRv6, the Binding SID sub-TLV does not have the ability to signal the SRv6 Endpoint Behavior [RFC8986] or the structure of the SID. It is RECOMMENDED that the SRv6 Binding SID TLV defined in Section 6.2, which enables the specification of the SRv6 Endpoint Behavior, be used for signaling of an SRv6 Binding SID.

6.2. SRv6 Binding SID TLV

The SRv6 Binding SID (BSID) is an optional TLV that is used to report the SRv6 BSID and its attributes for the SR Policy CP. The TLV MAY also optionally contain the Specified SRv6 BSID value for reporting as described in section 6.2.3 of [RFC9256]. Multiple instances of this TLV may be used to report each of the SRv6 BSIDs associated with the CP.

The TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           BSID Flags          |            RESERVED           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Binding SID (16 octets)                    //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Specified Binding SID (16 octets)             //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//   Sub-TLVs (variable)                                        //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 1212

* Length: variable

* BSID Flags: 2-octet field that indicates attribute and status of the Binding SID (BSID) associated with this CP. The following bit positions are defined and the semantics are described in detail in [RFC9256]. Other bits MUST be cleared by the originator and MUST be ignored by a receiver.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|B|U|F|                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- B-Flag: Indicates the allocation of the value in the BSID field when set and indicates that BSID is not allocated when clear.

- U-Flag: Indicates the specified BSID value is unavailable when set.

- F-Flag: Indicates the BSID value is one allocated dynamically due to fallback (e.g. when specified BSID is unavailable) when set.

* RESERVED: 2 octets. MUST be set to 0 by the originator and MUST be ignored by a receiver.

* Binding SID: It indicates the operational or allocated BSID value based on the status flags.

* Specified BSID: It is used to report the explicitly specified BSID value regardless of whether it is successfully allocated or not. The field is set to value 0 when BSID has not been specified.

* Sub-TLVs: variable and contains any other optional attributes associated with the SRv6 BSID.

The SRv6 Endpoint Behavior TLV (1250) and the SRv6 SID Structure TLV (1252) defined in [I-D.ietf-idr-bgpls-srv6-ext] are used as sub-TLVs of the SRv6 Binding SID TLV to optionally indicate the SRv6 Endpoint behavior and SID structure for the Binding SID value in the TLV.

6.3. SR Candidate Path State TLV

The SR Candidate Path (CP) State TLV provides the operational status and attributes of the SR Policy at the CP level. Only a single instance of this TLV is advertised for a given CP. If multiple instances are present, then the first one is considered valid and the rest are ignored.

The TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Priority    |   RESERVED    |              Flags            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Preference (4 octets)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 1202

* Length: 8 octets

* Priority: 1-octet value which indicates the priority of the CP. Refer Section 2.12 of [RFC9256].

* RESERVED: 1 octet. MUST be set to 0 by the originator and MUST be ignored by a receiver.

* Flags: 2-octet field that indicates attribute and status of the CP. The following bit positions are defined and the semantics are described in detail in [RFC9256]. Other bits MUST be cleared by the originator and MUST be ignored by a receiver.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|A|B|E|V|O|D|C|I|T|U|         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- S-Flag: Indicates the CP is in an administrative shut state when set

- A-Flag: Indicates the CP is the active path (i.e. one provisioned in the forwarding plane) for the SR Policy when set

- B-Flag: Indicates the CP is the backup path (i.e. one identified for path protection of the active path) for the SR Policy when set

- E-Flag: Indicates that the CP has been evaluated for validity (e.g. headend may evaluate CPs based on their preferences) when set

- V-Flag: Indicates the CP has at least one valid SID-List when set. When the E-Flag is clear (i.e. the CP has not been evaluated), then this flag MUST be set to 0 by the originator and ignored by the receiver.

- O-Flag: Indicates the CP was instantiated by the headend due to an on-demand nexthop trigger based on a local template when set. Refer to section 8.5 of [RFC9256] for details.

- D-Flag: Indicates the CP was delegated for computation to a PCE/controller when set

- C-Flag: Indicates the CP was provisioned by a PCE/controller when set

- I-Flag: Indicates the CP will perform the "drop upon invalid" behavior when no other active path is available for this SR Policy and this path is the one with the best preference amongst the available CPs. Refer to section 8.2 of [RFC9256] for details.

- T-Flag: Indicates the CP has been marked as eligible for use as Transit Policy on the headend when set. Refer to section 8.3 of [RFC9256].

- U-Flag: Indicates the SR Policy that the CP belongs to is dropping traffic as a result of the "drop upon invalid" behavior being activated.

* Preference: 4-octet value which indicates the preference of the CP. Refer to section 2.7 of [RFC9256] for details.

6.4. SR Policy Name TLV

The SR Policy Name TLV is an optional TLV that is used to carry the symbolic name associated with the SR Policy. Only a single instance of this TLV is advertised for a given CP. If multiple instances are present, then the first one is considered valid and the rest are ignored.

The TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Type            |              Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    SR Policy Name (variable)                  //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 1213

* Length: variable

* SR Policy Name: Symbolic name for the SR Policy without a NULL terminator as specified in section 2.6 of [RFC9256]. It is RECOMMENDED that the size of the symbolic name be limited to 255 bytes. Implementations MAY choose to truncate long names to 255 bytes when signaling via BGP-LS.

6.5. SR Candidate Path Name TLV

The SR Candidate Path Name TLV is an optional TLV that is used to carry the symbolic name associated with the candidate path. Only a single instance of this TLV is advertised for a given CP. If multiple instances are present, then the first one is considered valid and the rest are ignored.

The TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Type            |              Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Candidate Path Name (variable)                //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 1203

* Length: variable

* Candidate Path Name: Symbolic name for the SR Policy candidate path without a NULL terminator as specified in section 2.6 of [RFC9256]. It is RECOMMENDED that the size of the symbolic name be limited to 255 bytes. Implementations MAY choose to truncate long names to 255 bytes when signaling via BGP-LS.

6.6. SR Candidate Path Constraints TLV

The SR Candidate Path Constraints TLV is an optional TLV that is used to report the constraints associated with the candidate path. The constraints are generally applied to a dynamic candidate path which is computed either by the headend or may be delegated to a controller. The constraints may also be applied to an explicit path where the computation entity is expected to validate that the path satisfies the specified constraints and if not the path is to be invalidated (e.g., due to topology changes). Only a single instance of this TLV is advertised for a given CP. If multiple instances are present, then the first one is considered valid and the rest are

ignored.

The TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Type            |              Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Flags            |          RESERVED1            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             MTID              |   Algorithm   |   RESERVED2   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   sub-TLVs (variable)                                       //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 1204

* Length: variable

* Flags: 2-octet field that indicates the constraints that are being applied to the CP. The following bit positions are defined and the other bits MUST be cleared by the originator and MUST be ignored by a receiver.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|D|P|U|A|T|S|F|               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- D-Flag: Indicates that the CP uses SRv6 dataplane when set and SR/MPLS dataplane when clear

- P-Flag: Indicates that the CP prefers the use of only protected SIDs when set. This flag is mutually exclusive with the U-Flag.

- U-Flag: Indicates that the CP prefers the use of only unprotected SIDs when set. This flag is mutually exclusive with the P-Flag.

- A-Flag: Indicates that the CP uses only the SIDs belonging to the specified SR Algorithm when set

- T-Flag: Indicates that the CP uses only the SIDs belonging to the specified topology when set

- S-Flag: Indicates that the use of protected (P-Flag) or unprotected (U-Flag) SIDs becomes a strict constraint instead of a preference when set

- F-Flag: Indicates that the CP is fixed once computed and not modified except on operator intervention.

* RESERVED1: 2 octets. MUST be set to 0 by the originator and MUST be ignored by a receiver.

* MTID: Indicates the multi-topology identifier of the IGP topology that is preferred to be used when the path is set up. When the T-flag is set then the path is strictly using the specified topology SIDs only.

* Algorithm: Indicates the algorithm that is preferred to be used when the path is set up. When the A-flag is set then the path is strictly using the specified algorithm SIDs only.

* RESERVED2: 1 octet. MUST be set to 0 by the originator and MUST be ignored by a receiver.

* sub-TLVs: optional sub-TLVs MAY be included in this TLV to describe other constraints.

The following constraint sub-TLVs are defined for the SR CP Constraints TLV.

6.6.1. SR Affinity Constraint Sub-TLV

The SR Affinity Constraint sub-TLV is an optional sub-TLV of the SR CP Constraints TLV that is used to carry the affinity constraints [RFC2702] associated with the candidate path. The affinity is expressed in terms of Extended Admin Group (EAG) as defined in [RFC7308]. Only a single instance of this sub-TLV is advertised for a given CP. If multiple instances are present, then the first one is considered valid and the rest are ignored.

The sub-TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Type            |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Excl-Any-Size | Incl-Any-Size | Incl-All-Size |    RESERVED   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Exclude-Any EAG (optional, variable)              //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Include-Any EAG (optional, variable)              //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Include-All EAG (optional, variable)              //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 1208

* Length: variable, dependent on the size of the Extended Admin Group. MUST be a multiple of 4 octets.

* Exclude-Any-Size: one octet to indicate the size of Exclude-Any EAG bitmask size in multiples of 4 octets. (e.g. value 0 indicates the Exclude-Any EAG field is skipped, value 1 indicates that 4 octets of Exclude-Any EAG is included)

* Include-Any-Size: one octet to indicate the size of Include-Any EAG bitmask size in multiples of 4 octets. (e.g. value 0 indicates the Include-Any EAG field is skipped, value 1 indicates that 4 octets of Include-Any EAG is included)

* Include-All-Size: one octet to indicate the size of Include-All EAG bitmask size in multiples of 4 octets. (e.g. value 0 indicates the Include-All EAG field is skipped, value 1 indicates that 4 octets of Include-All EAG is included)

* RESERVED: 1 octet. MUST be set to 0 by the originator and MUST be ignored by a receiver.

* Exclude-Any EAG: the bitmask used to represent the affinities that have been excluded from the path.

* Include-Any EAG: the bitmask used to represent the affinities that have been included in the path.

* Include-All EAG: the bitmask used to represent all the affinities that have been included in the path.

6.6.2. SR SRLG Constraint Sub-TLV

The SR SRLG Constraint sub-TLV is an optional sub-TLV of the SR CP Constraints TLV that is used to carry the Shared Risk Link Group (SRLG) values [RFC4202] that have been excluded from the candidate path. Only a single instance of this sub-TLV is advertised for a given CP. If multiple instances are present, then the first one is considered valid and the rest are ignored.

The sub-TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Type            |              Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         SRLG Values (variable, multiples of 4 octets)        //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 1209

* Length: variable, dependent on the number of SRLGs encoded. MUST be a multiple of 4 octets.

* SRLG Values: One or more SRLG values (each of 4 octets).

6.6.3. SR Bandwidth Constraint Sub-TLV

The SR Bandwidth Constraint sub-TLV is an optional sub-TLV of the SR CP Constraints TLV that is used to indicate the bandwidth that has been requested for the candidate path. Only a single instance of this sub-TLV is advertised for a given CP. If multiple instances are present, then the first one is considered valid and the rest are ignored.

The sub-TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Type            |              Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Bandwidth                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 1210

* Length: 4 octets

* Bandwidth: 4 octets which specify the desired bandwidth in unit of bytes per second in IEEE floating point format.

6.6.4. SR Disjoint Group Constraint Sub-TLV

The SR Disjoint Group Constraint sub-TLV is an optional sub-TLV of the SR CP Constraints TLV that is used to carry the disjointness constraint associated with the candidate path. The disjointness between two SR Policy Candidate Paths is expressed by associating them with the same disjoint group identifier and then specifying the type of disjointness required between their paths. The computation is expected to achieve the highest level of disjointness requested and when that is not possible then fallback to a lesser level progressively based on the levels indicated. Only a single instance of this sub-TLV is advertised for a given CP. If multiple instances are present, then the first one is considered valid and the rest are ignored.

The sub-TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Type            |              Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Request-Flags |  Status-Flags |            RESERVED           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Disjoint Group Identifier                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 1211

* Length: 8 octets

* Request Flags: one octet to indicate the level of disjointness requested as specified in the form of flags. The following flags are defined and the other bits MUST be cleared by the originator and MUST be ignored by a receiver.

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|S|N|L|F|I|     |
+-+-+-+-+-+-+-+-+
```

where:

- S-Flag: Indicates that SRLG disjointness is requested when set

- N-Flag: Indicates that node disjointness is requested when set

- L-Flag: Indicates that link disjointness is requested when set

- F-Flag: Indicates that the computation may fallback to a lower level of disjointness amongst the ones requested when all cannot be achieved when set

- I-Flag: Indicates that the computation may fallback to the default best path (e.g. IGP path) in case of none of the desired disjointness can be achieved when set

* Status Flags: one octet to indicate the level of disjointness that has been achieved by the computation as specified in the form of flags. The following flags are defined and the other bits MUST be cleared by the originator and MUST be ignored by a receiver.

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|S|N|L|F|I|X|   |
+-+-+-+-+-+-+-+-+
```

where:

- S-Flag: Indicates that SRLG disjointness is achieved when set

- N-Flag: Indicates that node disjointness is achieved when set

- L-Flag: Indicates that link disjointness is achieved when set

- F-Flag: Indicates that the computation has fallen back to a lower level of disjointness than requested when set

- I-Flag: Indicates that the computation has fallen back to the best path (e.g. IGP path) and disjointness has not been achieved when set

- X-Flag : Indicates that the disjointness constraint could not be achieved and hence path has been invalidated when set

* RESERVED: 2 octets. MUST be set to 0 by the originator and MUST be ignored by a receiver.

* Disjointness Group Identifier: 4-octet value that is the group identifier for a set of disjoint paths

6.6.5. SR Bidirectional Group Constraint Sub-TLV

The SR Bidirectional Group Constraint sub-TLV is an optional sub-TLV of the SR CP Constraints TLV that is used to carry the bidirectional constraint associated with the candidate path. The bidirectional relationship between two SR Policy Candidate Paths is expressed by associating them with the same bidirectional group identifier and then specifying the type of bidirectional routing required between their paths. Only a single instance of this sub-TLV is advertised for a given CP. If multiple instances are present, then the first one is considered valid and the rest are ignored.

The sub-TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Type            |              Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Flags            |            RESERVED           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Bidirectional Group Identifier                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: TBD

* Length: 8 octets

* Flags: two octets to indicate the bidirectional path setup information as specified in the form of flags. The following flags are defined and the other bits MUST be cleared by the originator and MUST be ignored by a receiver.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|C|                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- R-Flag: Indicates that this CP of the SR Policy forms the reverse path when set and otherwise it is the forward path when clear

- C-Flag: Indicates that the bidirectional path is co-routed when set

* RESERVED: 2 octets. MUST be set to 0 by the originator and MUST be ignored by a receiver.

* Bidirectional Group Identifier: 4-octet value that is the group identifier for a set of bidirectional paths

6.6.6. SR Metric Constraint Sub-TLV

The SR Metric Constraint sub-TLV is an optional sub-TLV of the SR CP Constraints TLV that is used to report the optimization metric of the CP. For a dynamic path computation, it is used to report the optimization metric used along with its parameters. For an explicit path, this sub-TLV MAY be used to report the metric margin or bound to be used for validation (i.e., the path is invalidated if the metric is beyond specified values). Multiple instances of this sub-TLV may be used to report different metric type uses.

The sub-TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Metric Type  |      Flags    |          RESERVED             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Metric Margin                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Metric Bound                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: TBD

* Length: 12 octets

* Metric Type: 1-octet field which identifies the type of the metric being used. The metric type code points are listed in Section 9.8 of this document.

* Flags: 1-octet field that indicates the validity of the metric fields and their semantics. The following bit positions are defined and the other bits MUST be cleared by the originator and MUST be ignored by a receiver.

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|O|M|A|B|       |
+-+-+-+-+-+-+-+-+
```

where:

- O-Flag: Indicates that this is the optimization metric being reported for a dynamic CP when set. This bit MUST NOT be set in more than one instance of this TLV for a given CP advertisement.

- M-Flag: Indicates that the metric margin allowed is specified when set.

- A-Flag: Indicates that the metric margin is specified as an absolute value when set and is expressed as a percentage of the minimum metric when clear.

- B-Flag: Indicates that the metric bound allowed for the path is specified when set.

* RESERVED: 2 octets. MUST be set to 0 by the originator and MUST be ignored by a receiver.

* Metric Margin: 4-octet value which indicates the metric margin when the M-flag is set. The metric margin is specified as either an absolute value or as a percentage of the minimum computed path metric based on the A-flag. The metric margin loosens the criteria for minimum metric path calculation up to the specified metric to accommodate for other factors such as bandwidth availability, minimal SID stack depth, and maximizing of ECMP for the SR path computed.

* Metric Bound: 4-octet value which indicates the maximum metric that is allowed when the B-flag is set. If the computed path metric crosses the specified bound value then the path is considered invalid.

6.7. SR Segment List TLV

The SR Segment List TLV is used to report a single SID-List of a CP. Multiple instances of this TLV may be used to report multiple SID-Lists of a CP.

The TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Flags            |           RESERVED            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             MTID              |   Algorithm   |    RESERVED   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Weight (4 octets)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   sub-TLVs (variable)                                        //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 1205

* Length: variable

* Flags: 2-octet field that indicates attribute and status of the SID-List.The following bit positions are defined and the semantics are described in detail in [RFC9256]. Other bits MUST be cleared by the originator and MUST be ignored by a receiver.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|D|E|C|V|R|F|A|T|M|               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- D-Flag: Indicates the SID-List is comprised of SRv6 SIDs when set and indicates it is comprised of SR/MPLS labels when clear.

- E-Flag: Indicates that SID-List is an explicit path when set and indicates a dynamic path when clear.

- C-Flag: Indicates that SID-List has been computed for a dynamic path when set. It is always reported as set for explicit paths.

- V-Flag: Indicates the SID-List has passed verification or its verification was not required when set and failed verification when clear.

- R-Flag: Indicates that the first Segment has been resolved when set and failed resolution when clear.

- F-Flag: Indicates that the computation for the dynamic path failed when set and succeeded (or not required in case of explicit path) when clear

- A-Flag: Indicates that all the SIDs in the SID-List belong to the specified algorithm when set.

- T-Flag: Indicates that all the SIDs in the SID-List belong to the specified topology (identified by the multi-topology ID) when set.

- M-Flag: Indicates that the SID-list has been removed from the forwarding plane due to fault detection by a monitoring mechanism (e.g. BFD) when set and indicates no fault detected or monitoring is not being done when clear.

* RESERVED: 2 octets. MUST be set to 0 by the originator and MUST be ignored by a receiver.

* MTID: 2 octets that indicates the multi-topology identifier of the IGP topology that is to be used when the T-flag is set.

* Algorithm: 1 octet that indicates the algorithm of the SIDs used in the SID-List when the A-flag is set.

* RESERVED: 1 octet. MUST be set to 0 by the originator and MUST be ignored by a receiver.

* Weight: 4-octet field that indicates the weight associated with the SID-List for weighted load-balancing. Refer to section 2.2 and 2.11 of [RFC9256].

* Sub-TLVs: variable and contains the ordered set of Segments and any other optional attributes associated with the specific SID-List.

The SR Segment sub-TLV (defined in Section 6.8) MUST be included as an ordered set of sub-TLVs within the SR Segment List TLV when the SID-List is not empty. A SID-List may be empty in certain cases (e.g. for a dynamic path) where the headend has not yet performed the computation and hence not derived the segments required for the path; in such cases, the SR Segment List TLV SHOULD NOT include any SR Segment sub-TLVs.

6.8. SR Segment Sub-TLV

The SR Segment sub-TLV describes a single segment in a SID-List. One or more instances of this sub-TLV in an ordered manner constitute a SID-List for an SR Policy candidate path. It is a sub-TLV of the SR Segment List TLV and it has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Segment Type  |    RESERVED   |             Flags             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    SID (4 or 16 octets)                       //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//               Segment Descriptor (variable)                 //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//   Sub-TLVs (variable)                                        //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 1206

* Length: variable

* Segment Type: 1 octet which indicates the type of segment (refer Section 6.8.1 for details)

* RESERVED: 1 octet. MUST be set to 0 by the originator and MUST be ignored by a receiver.

* Flags: 2-octet field that indicates attribute and status of the Segment and its SID. The following bit positions are defined and the semantics are described in detail in [RFC9256]. Other bits MUST be cleared by the originator and MUST be ignored by a receiver.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|E|V|R|A|                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- S-Flag: Indicates the presence of SID value in the SID field when set and that no value is indicated when clear.

- E-Flag: Indicates the SID value is explicitly provisioned value (locally on headend or via controller/PCE) when set and is a dynamically resolved value by headend when clear

- V-Flag: Indicates the SID has passed verification or did not require verification when set and failed verification when clear.

- R-Flag: Indicates the SID has been resolved or did not require resolution (e.g. because it is not the first SID) when set and failed resolution when clear.

- A-Flag: Indicates that the Algorithm indicated in the Segment descriptor is valid when set. When clear, it indicates that the headend is unable to determine the algorithm of the SID.

* SID: 4 octets carrying the MPLS Label or 16 octets carrying the SRv6 SID based on the Segment Type. When carrying the MPLS Label, as shown in the figure below, the TC, S, and TTL (total of 12 bits) are RESERVED and MUST be set to 0 by the originator and MUST be ignored by a receiver.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Label                        | TC  |S|       TTL     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* Segment Descriptor: variable size Segment descriptor based on the type of segment (refer to Section 6.8.1 for details)

* Sub-Sub-TLVs: variable and contains any other optional attributes associated with the specific segment.

The SRv6 Endpoint Behavior TLV (1250) and the SRv6 SID Structure TLV (1252) defined in [I-D.ietf-idr-bgpls-srv6-ext] are used as sub-sub-TLVs of the SR Segment sub-TLV to optionally indicate the SRv6 Endpoint behavior and SID structure when advertising the SRv6 specific segment types.

6.8.1. Segment Descriptors

Section 4 of [RFC9256] defines multiple types of segments and their description. This section defines the encoding of the Segment Descriptors for each of those Segment types to be used in the Segment sub-TLV describes previously in Section 6.8.

The following types are currently defined and their mapping to the respective segment types defined in [RFC9256]:

Type	Segment Description
1	(Type A) SR-MPLS Label
2	(Type B) SRv6 SID as IPv6 address
3	(Type C) SR-MPLS Prefix SID as IPv4 Node Address
4	(Type D) SR-MPLS Prefix SID as IPv6 Node Global Address
5	(Type E) SR-MPLS Adjacency SID as IPv4 Node Address & Local Interface ID
6	(Type F) SR-MPLS Adjacency SID as IPv4 Local & Remote Interface Addresses
7	(Type G) SR-MPLS Adjacency SID as pair of IPv6 Global Address & Interface ID for Local & Remote nodes
8	(Type H) SR-MPLS Adjacency SID as pair of IPv6 Global Addresses for the Local & Remote Interface
9	(Type I) SRv6 END SID as IPv6 Node Global Address
10	(Type J) SRv6 END.X SID as pair of IPv6 Global Address & Interface ID for Local & Remote nodes
11	(Type K) SRv6 END.X SID as pair of IPv6 Global Addresses for the Local & Remote Interface

6.8.1.1. Type 1: SR-MPLS Label

The Segment is SR-MPLS type and is specified simply as the label. The format of its Segment Descriptor is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|   Algorithm   |
+-+-+-+-+-+-+-+-+
```

Where:

* Algorithm: 1-octet value that indicates the algorithm used for picking the SID. This is valid only when the A-flag has been set in the Segment TLV.

6.8.1.2. Type 2: SRv6 SID

The Segment is SRv6 type and is specified simply as the SRv6 SID address. The format of its Segment Descriptor is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|   Algorithm   |
+-+-+-+-+-+-+-+-+
```

Where:

* Algorithm: 1-octet value that indicates the algorithm used for picking the SID. This is valid only when the A-flag has been set in the Segment TLV.

6.8.1.3. Type 3: SR-MPLS Prefix SID for IPv4

The Segment is SR-MPLS Prefix SID type and is specified as an IPv4 node address. The format of its Segment Descriptor is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|   Algorithm   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 IPv4 Node Address (4 octets)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:

* Algorithm: 1-octet value that indicates the algorithm used for picking the SID

* IPv4 Node Address: 4-octet value which carries the IPv4 address associated with the node

6.8.1.4. Type 4: SR-MPLS Prefix SID for IPv6

The Segment is SR-MPLS Prefix SID type and is specified as an IPv6 global address. The format of its Segment Descriptor is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|   Algorithm   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          IPv6 Node Global Address (16 octets)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:

* Algorithm: 1-octet value that indicates the algorithm used for picking the SID

* IPv6 Node Global Address: 16-octet value which carries the IPv6 global address associated with the node

6.8.1.5. Type 5: SR-MPLS Adjacency SID for IPv4 with an Interface ID

The Segment is SR-MPLS Adjacency SID type and is specified as an IPv4 node address along with the local interface ID on that node. The format of its Segment Descriptor is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 IPv4 Node Address (4 octets)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Local Interface ID (4 octets)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:

* IPv4 Node Address: 4-octet value which carries the IPv4 address associated with the node

* Local Interface ID: 4-octet value which carries the local interface ID of the node identified by the Node Address

6.8.1.6. Type 6: SR-MPLS Adjacency SID for IPv4 with an Interface Address

The Segment is SR-MPLS Adjacency SID type and is specified as a pair of IPv4 local and remote addresses. The format of its Segment Descriptor is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                IPv4 Local Address (4 octets)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               IPv4 Remote Address (4 octets)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:

* IPv4 Local Address: 4-octet value which carries the local IPv4 address associated with the node

* IPv4 Remote Address: 4-octet value which carries the remote IPv4 address associated with the node's neighbor. This is optional and MAY be set to 0 when not used (e.g. when identifying point-to-point links).

6.8.1.7. Type 7: SR-MPLS Adjacency SID for IPv6 with an interface ID

The Segment is SR-MPLS Adjacency SID type and is specified as a pair of IPv6 global address and interface ID for local and remote nodes. The format of its Segment Descriptor is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          IPv6 Local Node Global Address (16 octets)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Local Node Interface ID (4 octets)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          IPv6 Remote Node Global Address (16 octets)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Remote Node Interface ID (4 octets)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:

* IPv6 Local Node Global Address: 16-octet value which carries the IPv6 global address associated with the local node

* Local Node Interface ID : 4-octet value which carries the interface ID of the local node identified by the Local Node Address

* IPv6 Remote Node Global Address: 16-octet value which carries the IPv6 global address associated with the remote node. This is optional and MAY be set to 0 when not used (e.g. when identifying point-to-point links).

* Remote Node Interface ID: 4-octet value which carries the interface ID of the remote node identified by the Remote Node Address. This is optional and MAY be set to 0 when not used (e.g. when identifying point-to-point links).

6.8.1.8. Type 8: SR-MPLS Adjacency SID for IPv6 with an Interface Address

The Segment is SR-MPLS Adjacency SID type and is specified as a pair of IPv6 Global addresses for local and remote interface addresses. The format of its Segment Descriptor is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Global IPv6 Local Interface Address (16 octets)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Global IPv6 Remote Interface Address (16 octets)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:

* IPv6 Local Address: 16-octet value which carries the local IPv6 address associated with the node

* IPv6 Remote Address: 16-octet value which carries the remote IPv6 address associated with the node’s neighbor

6.8.1.9. Type 9: SRv6 END SID as IPv6 Node Address

The Segment is SRv6 END SID type and is specified as an IPv6 global address. The format of its Segment Descriptor is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|   Algorithm   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          IPv6 Node Global Address (16 octets)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:

* Algorithm: 1-octet value that indicates the algorithm used for picking the SID

* IPv6 Node Global Address: 16-octet value which carries the IPv6 global address associated with the node

6.8.1.10. Type 10: SRv6 END.X SID as an Anterface ID

The Segment is SRv6 END.X SID type and is specified as a pair of IPv6 global address and interface ID for local and remote nodes. The format of its Segment Descriptor is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          IPv6 Local Node Global Address (16 octets)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Local Node Interface ID (4 octets)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          IPv6 Remote Node Global Address (16 octets)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Remote Node Interface ID (4 octets)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:

* IPv6 Local Node Global Address: 16-octet value which carries the IPv6 global address associated with the local node

* Local Node Interface ID: 4-octet value which carries the interface ID of the local node identified by the Local Node Address

* IPv6 Remote Node Global Address: 16-octet value which carries the IPv6 global address associated with the remote node. This is optional and MAY be set to 0 when not used (e.g. when identifying point-to-point links).

* Remote Node Interface ID: 4-octet value which carries the interface ID of the remote node identified by the Remote Node Address. This is optional and MAY be set to 0 when not used (e.g. when identifying point-to-point links).

6.8.1.11. Type 11: SRv6 END.X SID as an Interface Address

The Segment is SRv6 END.X SID type and is specified as a pair of IPv6 Global addresses for local and remote interface addresses. The format of its Segment Descriptor is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Global IPv6 Local Interface Address (16 octets)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Global IPv6 Remote Interface Address (16 octets)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:

* IPv6 Local Address: 16-octet value which carries the local IPv6 address associated with the node

* IPv6 Remote Address: 16-octet value which carries the remote IPv6 address associated with the node’s neighbor

6.9. SR Segment List Metric Sub-TLV

The SR Segment List Metric sub-TLV reports the computed metric of the specific SID-List. It is used to report the type of metric and its computed value by the computation entity (i.e., either the headend or the controller when the path is delegated) when available. More than one instance of this sub-TLV may be present in SR Segment List to report metric values of different metric types. The metric margin and bound may be optionally reported using this sub-TLV when this information is not being reported using the SR Metric Constraint sub-TLV (refer to Section 6.6.6) at the SR CP level.

It is a sub-TLV of the SR Segment List TLV and has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Metric Type  |      Flags    |          RESERVED             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Metric Margin                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Metric Bound                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Metric Value                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: 1207

* Length: 16 octets

* Metric Type: 1-octet field which identifies the type of metric. The metric type code points are listed in Section 9.8 of this document.

* Flags: 1-octet field that indicates the validity of the metric fields and their semantics. The following bit positions are defined and the other bits MUST be cleared by the originator and MUST be ignored by a receiver.

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|M|A|B|V|       |
+-+-+-+-+-+-+-+-+
```

where:

- M-Flag: Indicates that the metric margin allowed for this path computation is specified when set

- A-Flag: Indicates that the metric margin is specified as an absolute value when set and is expressed as a percentage of the minimum metric when clear.

- B-Flag: Indicates that the metric bound allowed for the path is specified when set.

- V-Flag: Indicates that the metric value computed is being reported when set.

* RESERVED: 2 octets. MUST be set to 0 by the originator and MUST be ignored by a receiver.

* Metric Margin: 4-octet value which indicates the metric margin value when the M-flag is set. The metric margin is specified as either an absolute value or as a percentage of the minimum computed path metric based on the A-flag. The metric margin loosens the criteria for minimum metric path calculation up to the specified metric to accomodate for other factors such as bandwidth availability, minimal SID stack depth, and maximizing of ECMP for the SR path computed.

* Metric Bound: 4-octet value which indicates the maximum metric value that is allowed when the B-flag is set. If the computed path metric crosses the specified bound value then the path is considered invalid.

* Metric Value: 4-octet value which indicates the metric of the computed path when the V-flag is set. This value is available and reported when the computation is successful and a valid path is available.

6.10. SR Segment List Bandwidth Sub-TLV

The SR Segment List Bandwidth sub-TLV is an optional sub-TLV used to report the bandwidth allocated to the specific SID-List by the path computation entity. Only a single instance of this sub-TLV is advertised for a given Segment List. If multiple instances are present, then the first one is considered valid and the rest are ignored.

It is a sub-TLV of the SR Segment List TLV and has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Bandwidth                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* Type: TBD

* Length: 4 octets

* Bandwidth: 4 octets which specify the allocated bandwidth in unit of bytes per second in IEEE floating point format.

7. Procedures

The BGP-LS advertisements for the TE policy NLRI types are originated by the headend node for the TE Policies that are instantiated on its local node.

For MPLS TE LSPs signaled via RSVP-TE, the NLRI descriptor TLVs as specified in Section 4.1, Section 4.2, Section 4.3, and Section 4.4 are used. Then the TE LSP state is encoded in the BGP-LS Attribute field as MPLS-TE Policy State TLV as described in Section 5. The RSVP-TE objects that reflect the state of the LSP are included as defined in Section 5.1. When the TE LSP is setup with the help of PCEP signaling then another MPLS-TE Policy State TLV SHOULD be used to encode the related PCEP objects corresponding to the LSP as defined in Section 5.2.

For the reporting of SR Policy Candidate Paths, the NLRI descriptor TLV as specified in Section 4.5 is used. An SR Policy candidate path (CP) may be instantiated on the headend node via a local configuration, PCEP, or BGP SR Policy signaling and this is indicated via the SR Protocol Origin. Then the SR Policy Candidate Path's state and attributes are encoded in the BGP-LS Attribute field as SR Policy State TLVs and sub-TLVs as described in Section 6. The SR Candidate Path State TLV as defined in Section 6.3 is included to report the state of the CP. The SR BSID TLV as defined in Section 6.1 or Section 6.2 is included to report the BSID of the CP when one is either specified or allocated by the headend. The constraints and the optimization metric for the SR Policy Candidate Path are reported using the SR Candidate Path Constraints TLV and its sub-TLVs as described in Section 6.6. The SR Segment List TLV is included for each of the SID-List(s) associated with the CP. Each SR Segment List TLV in turn includes SR Segment sub-TLV(s) to report the segment(s) and their status. The SR Segment List Metric sub-TLV is used to report the metric values at an individual SID List level.

When the SR Policy CP is setup with the help of PCEP signaling then another MPLS-TE Policy State TLV MAY be used to encode the related PCEP objects corresponding to the LSP as defined in Section 5.2 specifically to report information and status that is not covered by the defined TLVs under Section 6. In the event of a conflict of information, the receiver MUST prefer the information originated via TLVs defined in Section 6 over the PCEP objects reported via the TE Policy State TLV.

8. Manageability Considerations

The Existing BGP operational and management procedures apply to this document. No new procedures are defined in this document. The considerations as specified in [RFC7752] apply to this document.

In general, it is assumed that the TE Policy head-end nodes are responsible for the distribution of TE Policy state information, while other nodes, e.g. the nodes in the path of a policy, MAY report the TE Policy information (if available) when needed. For example, the border routers in the inter-domain case will also distribute LSP state information since the ingress node may not have the complete information for the end-to-end path.

9. IANA Considerations

This section describes the code point allocation by IANA for this document.

9.1. BGP-LS NLRI-Types

IANA maintains a registry called "BGP-LS NLRI-Types" in the "Border Gateway Protocol - Link State (BGP-LS) Parameters" registry group.

The following table lists the status of code points that have been allocated by IANA and others that are pending allocation:

Type	NLRI Type	Reference
5	SR Policy Candidate Path NLRI	this document
TBD	MPLS-TE LSP NLRI	this document
TBD	MPLS Local Cross-connect NLRI	this document

9.2. BGP-LS Protocol-IDs

IANA maintains a registry called "BGP-LS Protocol-IDs" in the "Border Gateway Protocol - Link State (BGP-LS) Parameters" registry group.

The following Protocol-ID codepoints have been allocated by IANA:

Protocol-ID	NLRI information source protocol	Reference
8	RSVP-TE	this document
9	Segment Routing	this document

9.3. BGP-LS TLVs

IANA maintains a registry called "Node Anchor, Link Descriptor and Link Attribute TLVs" in the "Border Gateway Protocol - Link State (BGP-LS) Parameters" registry group.

The following table lists the status of TLV code points that have been allocated by IANA and others that are pending allocation:

Code Point	Description	Value defined in
550	Tunnel ID	this document
551	LSP ID	this document
552	IPv4/6 Tunnel Head-end address	this document
553	IPv4/6 Tunnel Tail-end address	this document
554	SR Policy CP Descriptor	this document
555	MPLS Local Cross Connect	this document
556	MPLS Cross Connect Interface	this document
557	MPLS Cross Connect FEC	this document
1200	MPLS-TE Policy State	this document
1201	SR Binding SID	this document
1202	SR CP State	this document
1203	SR CP Name	this document
1204	SR CP Constraints	this document
1205	SR Segment List	this document
1206	SR Segment	this document
1207	SR Segment List Metric	this document
1208	SR Affinity Constraint	this document
1209	SR SRLG Constraint	this document
1210	SR Bandwidth Constraint	this document
1211	SR Disjoint Group Constraint	this document
1212	SRv6 Binding SID	this document
1213	SR Policy Name	this document
TBD	SR Bidirectional Group Constraint	this document
TBD	SR Metric Constraint	this document
TBD	SR Segment List Bandwidth	this document

9.4. BGP-LS SR Policy Protocol Origin

This document requests IANA to maintain a new registry under "Border Gateway Protocol - Link State (BGP-LS) Parameters" registry group with the allocation policy of "Expert Review" [RFC8126] using the guidelines for Designated Experts as specified in [RFC9029]. The new registry is called "SR Policy Protocol Origin" and contains the codepoints allocated to the "Protocol Origin" field defined in

Section 4.5. The registry contains the following codepoints, with initial values, to be assigned by IANA with the reference set to this document:

Code Point	Protocol Origin
0	Reserved (not to be used)
1	PCEP
2	BGP SR Policy
3	Configuration (CLI, YANG model via NETCONF, etc.)
4-250	Unassigned
251-255	Private Use (not to be assigned by IANA)

9.5. BGP-LS TE State Object Origin

This document requests IANA to maintain a new registry under "Border Gateway Protocol - Link State (BGP-LS) Parameters" registry group with the allocation policy of "Expert Review" [RFC8126] using the guidelines for Designated Experts as specified in [RFC9029]. The new registry is called "TE State Path Origin" and contains the codepoints allocated to the "Object Origin" field defined in Section 5. The registry contains the following codepoints, with initial values, to be assigned by IANA with the reference set to this document:

Code Point	Object Origin
0	Reserved (not to be used)
1	RSVP-TE
2	PCEP
3	Local/Static
4-250	Unassigned
251-255	Private Use (not to be assigned by IANA)

9.6. BGP-LS TE State Address Family

This document requests IANA to maintain a new registry under "Border Gateway Protocol - Link State (BGP-LS) Parameters" registry group with the allocation policy of "Expert Review" [RFC8126] using the guidelines for Designated Experts as specified in [RFC9029]. The new registry is called "TE State Address Family" and contains the codepoints allocated to the "Address Family" field defined in Section 5. The registry contains the following codepoints, with

initial values, to be assigned by IANA with the reference set to this document:

Code Point	Address Family
0	Reserved (not to be used)
1	MPLS-IPv4
2	MPLS-IPv6
3-250	Unassigned
251-255	Private Use (not to be assigned by IANA)

9.7. BGP-LS SR Segment Descriptors

This document requests IANA to maintain a new registry under "Border Gateway Protocol - Link State (BGP-LS) Parameters" registry group with the allocation policy of "Expert Review" [RFC8126] using the guidelines for Designated Experts as specified in [RFC9029]. The new registry is called "SR Segment Descriptor Types" and contains the codepoints allocated to the "Segment Type" field defined in Section 6.8 and described in Section 6.8.1. The registry contains the following codepoints, with initial values, to be assigned by IANA with the reference set to this document:

Code Point	Segment Description
0	Reserved (not to be used)
1	(Type A) SR-MPLS Label
2	(Type B) SRv6 SID as IPv6 address
3	(Type C) SR-MPLS Prefix SID as IPv4 Node Address
4	(Type D) SR-MPLS Prefix SID as IPv6 Node Global Address
5	(Type E) SR-MPLS Adjacency SID as IPv4 Node Address & Local Interface ID
6	(Type F) SR-MPLS Adjacency SID as IPv4 Local & Remote Interface Addresses
7	(Type G) SR-MPLS Adjacency SID as pair of IPv6 Global Address & Interface ID for Local & Remote nodes
8	(Type H) SR-MPLS Adjacency SID as pair of IPv6 Global Addresses for the Local & Remote Interface
9	(Type I) SRv6 END SID as IPv6 Node Global Address
10	(Type J) SRv6 END.X SID as pair of IPv6 Global Address & Interface ID for Local & Remote nodes
11	(Type K) SRv6 END.X SID as pair of IPv6 Global Addresses for the Local & Remote Interface
12-255	Unassigned

9.8. BGP-LS Metric Type

This document requests IANA to maintain a new registry under "Border Gateway Protocol - Link State (BGP-LS) Parameters" registry group with the allocation policy of "Expert Review" [RFC8126] using the guidelines for Designated Experts as specified in [RFC9029]. The new registry is called "Metric Type" and contains the codepoints allocated to the "metric type" field defined in Section 6.9. The registry contains the following codepoints, with initial values, to be assigned by IANA with the reference set to this document:

Code Point	Metric Type
0	IGP Metric
1	Min Unidirectional Link Delay [RFC7471]
2	TE Metric [RFC3630]
3	Hop Count (refer [RFC5440])
4	SID List Length
5-250	Unassigned
251-255	Private Use (not to be assigned by IANA)

10. Security Considerations

Procedures and protocol extensions defined in this document do not affect the BGP security model. See [RFC6952] for details.

11. Contributors

The following people have substantially contributed to the editing of this document:

Clarence Filsfils
Cisco Systems
Email: cfilsfil@cisco.com

12. Acknowledgements

The authors would like to thank Dhruv Dhody, Mohammed Abdul Aziz Khalid, Lou Berger, Acee Lindem, Siva Sivabalan, Arjun Sreekantiah, Dhanendra Jain, Francois Clad, Zafar Ali, Stephane Litkowski, and Aravind Babu Mahendra Babu for their review and valuable comments.

13. References

13.1. Normative References

[I-D.ietf-idr-bgpls-srv6-ext]
Dawra, G., Filsfils, C., Talaulikar, K., Chen, M., Bernier, D., and B. Decraene, "BGP Link State Extensions for SRv6", Work in Progress, Internet-Draft, draft-ietf-idr-bgpls-srv6-ext-13, 14 January 2023, <https://www.ietf.org/archive/id/draft-ietf-idr-bgpls-srv6-ext-13.txt>.

[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>.

[RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <https://www.rfc-editor.org/info/rfc2205>.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <https://www.rfc-editor.org/info/rfc3209>.

[RFC3473] Berger, L., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, DOI 10.17487/RFC3473, January 2003, <https://www.rfc-editor.org/info/rfc3473>.

[RFC4090] Pan, P., Ed., Swallow, G., Ed., and A. Atlas, Ed., "Fast Reroute Extensions to RSVP-TE for LSP Tunnels", RFC 4090, DOI 10.17487/RFC4090, May 2005, <https://www.rfc-editor.org/info/rfc4090>.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <https://www.rfc-editor.org/info/rfc4760>.

[RFC4872] Lang, J.P., Ed., Rekhter, Y., Ed., and D. Papadimitriou, Ed., "RSVP-TE Extensions in Support of End-to-End Generalized Multi-Protocol Label Switching (GMPLS) Recovery", RFC 4872, DOI 10.17487/RFC4872, May 2007, <https://www.rfc-editor.org/info/rfc4872>.

[RFC4873] Berger, L., Bryskin, I., Papadimitriou, D., and A. Farrel, "GMPLS Segment Recovery", RFC 4873, DOI 10.17487/RFC4873, May 2007, <https://www.rfc-editor.org/info/rfc4873>.

[RFC4874] Lee, CY., Farrel, A., and S. De Cnodder, "Exclude Routes - Extension to Resource ReserVation Protocol-Traffic Engineering (RSVP-TE)", RFC 4874, DOI 10.17487/RFC4874, April 2007, <https://www.rfc-editor.org/info/rfc4874>.

[RFC5420] Farrel, A., Ed., Papadimitriou, D., Vasseur, JP., and A. Ayyangar, "Encoding of Attributes for MPLS LSP Establishment Using Resource Reservation Protocol Traffic Engineering (RSVP-TE)", RFC 5420, DOI 10.17487/RFC5420, February 2009, <https://www.rfc-editor.org/info/rfc5420>.

[RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <https://www.rfc-editor.org/info/rfc5440>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

[RFC9029] Farrel, A., "Updates to the Allocation Policy for the Border Gateway Protocol - Link State (BGP-LS) Parameters Registries", RFC 9029, DOI 10.17487/RFC9029, June 2021, <https://www.rfc-editor.org/info/rfc9029>.

[RFC9086] Previdi, S., Talaulikar, K., Ed., Filsfils, C., Patel, K., Ray, S., and J. Dong, "Border Gateway Protocol - Link State (BGP-LS) Extensions for Segment Routing BGP Egress Peer Engineering", RFC 9086, DOI 10.17487/RFC9086, August 2021, <https://www.rfc-editor.org/info/rfc9086>.

[RFC9256] Filsfils, C., Talaulikar, K., Ed., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", RFC 9256, DOI 10.17487/RFC9256, July 2022, <https://www.rfc-editor.org/info/rfc9256>.

13.2. Informative References

[RFC2702] Awduche, D., Malcolm, J., Agogbua, J., O'Dell, M., and J. McManus, "Requirements for Traffic Engineering Over MPLS", RFC 2702, DOI 10.17487/RFC2702, September 1999, <https://www.rfc-editor.org/info/rfc2702>.

[RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <https://www.rfc-editor.org/info/rfc3630>.

[RFC4202] Kompella, K., Ed. and Y. Rekhter, Ed., "Routing Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)", RFC 4202, DOI 10.17487/RFC4202, October 2005, <https://www.rfc-editor.org/info/rfc4202>.

[RFC4655] Farrel, A., Vasseur, J.-P., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <https://www.rfc-editor.org/info/rfc4655>.

[RFC5065] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", RFC 5065, DOI 10.17487/RFC5065, August 2007, <https://www.rfc-editor.org/info/rfc5065>.

[RFC6952] Jethanandani, M., Patel, K., and L. Zheng, "Analysis of BGP, LDP, PCEP, and MSDP Issues According to the Keying and Authentication for Routing Protocols (KARP) Design Guide", RFC 6952, DOI 10.17487/RFC6952, May 2013, <https://www.rfc-editor.org/info/rfc6952>.

[RFC7308] Osborne, E., "Extended Administrative Groups in MPLS Traffic Engineering (MPLS-TE)", RFC 7308, DOI 10.17487/RFC7308, July 2014, <https://www.rfc-editor.org/info/rfc7308>.

[RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015, <https://www.rfc-editor.org/info/rfc7471>.

[RFC8231] Crabbe, E., Minei, I., Medved, J., and R. Varga, "Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE", RFC 8231, DOI 10.17487/RFC8231, September 2017, <https://www.rfc-editor.org/info/rfc8231>.

Authors' Addresses

Stefano Previdi
Email: stefano@previdi.net

Ketan Talaulikar (editor)
Cisco Systems
India
Email: ketant.ietf@gmail.com

Jie Dong (editor)
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing
100095
China
Email: jie.dong@huawei.com

Mach(Guoyi) Chen
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing
100095
China
Email: mach.chen@huawei.com

Hannes Gredler
RtBrick Inc.
Email: hannes@rtbrick.com

Jeff Tantsura
Microsoft
Email: jefftant.ietf@gmail.com

Inter-Domain Routing
Internet-Draft
Intended status: Standards Track
Expires: August 26, 2019

K. Talaulikar
P. Psenak
Cisco Systems
S. Zandi
G. Dawra
LinkedIn

February 22, 2019

Flexible Algorithm Definition Advertisement with BGP Link-State
draft-ketant-idr-bgp-ls-flex-algo-01

Abstract

Flexible Algorithm is a solution that allows routing protocols (viz. OSPF and IS-IS) to compute paths over a network based on user-defined (and hence, flexible) constraints and metrics. The computation is performed by routers participating in the specific network in a distribute manner using a Flex Algorithm definition. This definition provisioned on one or more routers and propagated (viz. OSPF and IS-IS flooding) through the network.

BGP Link-State (BGP-LS) enables the collection of various topology information from the network. This draft defines extensions to BGP-LS address-family to advertise the Flexible Algorithm Definition as a part of the topology information from the network.

Requirements Language

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.

Status of This Memo

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 August 26, 2019.

Copyright Notice

Table of Contents

1. Introduction

IGP protocols (OSPF and IS-IS) traditionally compute best paths over the network based on the IGP metric assigned to the links. Many network deployments use RSVP-TE [RFC3209] based or Segment Routing (SR) Policy [I-D.ietf-spring-segment-routing-policy] based solutions to enforce traffic over a path that is computed using different metrics or constraints than the shortest IGP path. [I-D.ietf-lsr-flex-algo] defines the Flexible Algorithm solution that

allows IGPs themselves to compute constraint based paths over the network.

Flexible Algorithm is called so as it allows a user the flexibility to define

- o the type of calculation to be used (e.g. shortest path)
- o the metric type to be used (e.g. IGP metric or TE metric)
- o the set of constraints to be used (e.g. inclusion or exclusion of certain links using affinities)

The operations of the flexible algorithm solution is described in detail in [I-D.ietf-lsr-flex-algo] and a high level summary of the same is described here for clarity. The network operator enables the participation of specific nodes in the network for a specific algorithm and then provisions the definition of that flexible algorithm on one or more of these nodes. The nodes where the flexible algorithm definition is advertised then flood these definitions via respective IGP (IS-IS and OSPFv2/v3) mechanisms to all other nodes in the network. The nodes select the definition for each algorithm based on the flooded information in a deterministic manner and thus all nodes participating in a flexible algorithm computation arrive at a common understanding of the type of calculation that they need to use.

When using Segment Routing (SR) [RFC8402] forwarding plane, the result of a flex algorithm computation is the provisioning of the Prefix SIDs associated with that algorithm with paths based on the topology computed based on that algorithm. This flex algorithm computation is within an IGP area or level similar to the default shortest path tree (SPT) algorithm.

The BGP-LS extensions for SR are defined in [I-D.ietf-idr-bgp-ls-segment-routing-ext] and includes the

- o SR Algorithm TLV to indicate the participation of a node in a flex algorithm computation
- o Prefix SID TLV to indicate the association of the Prefix-SIDs to a specific flex algorithm

Thus a controller or a Path Computation Engine (PCE) is aware of the IGP topology across multiple domains which includes the above information related to the flexible algorithm. This draft defines extensions to BGP-LS for carrying the Flexible Algorithm Definition information so that it enables the controller/PCE to learn the

mapping of the flex algorithm number to its definition in each area/ domain of the underlying IGP. The controller/PCE also learns the type of computation used and the constraints for the same. This information can then be leveraged by it for setting up SR Policy paths end to end across domains by leveraging the appropriate Flex Algorithm specific Prefix SIDs in its Segment List [I-D.ietf-spring-segment-routing-policy]. e.g. picking the Flex Algorithm Prefix SID or ABRs/ASBRs corresponding to a definition that optimizes on the delay metric enables the PCE/controller to build an end to end low latency path across IGP domains with minimal Prefix-SIDs in the SID list.

2. BGP-LS Extensions for Flex Algo Definition

The BGP-LS [RFC7752] specifies the Node NLRI for advertisement of nodes and their attributes using the BGP-LS Attribute. The Flexible Algorithm Definition (FAD) advertised by a node are considered as its node level attributes and advertised as such.

This document defines a new BGP-LS Attribute TLV called the Flexible Algorithm Definition (FAD) TLV and its format is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Flex-Algorithm |   Metric-Type |   Calc-Type   |    Priority   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                sub-TLVs       ...                            //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Flex Algorithm Definition TLV

where:

- o Type: TBD (see IANA Considerations Section 3)
- o Length: variable. Minimum of 8 octets.
- o Flex-Algorithm : 1 octet value in the range between 128 and 255 inclusive which is the range defined for Flexible Algorithms in the IANA "IGP Parameters" registries under the "IGP Algorithm Types" registry [I-D.ietf-lsr-flex-algo].
- o Metric-Type : 1 octet value indicating the type of the metric used in the computation. Values allowed come from the IANA "IGP

Parameters" registries under the "Flexible Algorithm Definition Metric-Type" registry [I-D.ietf-lsr-flex-algo].

- o Calculation-Type : 1 octet value in the range between 0 and 127 inclusive which is the range defined for the standard algorithms in the IANA "IGP Parameters" registries under the "IGP Algorithm Types" registry [I-D.ietf-lsr-flex-algo].

- o Priority : 1 octet value between 0 and 255 inclusive that specifies the priority of the FAD.

- o sub-TLVs : zero or more sub-TLVs may be included as described further in this section.

The FAD TLV can only be added to the BGP-LS Attribute of the Node NLRI if the corresponding node originates the underlying IGP TLV/sub-TLV as described below. This information is derived from the protocol specific advertisements as below..

- o IS-IS, as defined by the ISIS Flexible Algorithm Definition sub-TLV in [I-D.ietf-lsr-flex-algo].

- o OSPFv2/OSPFv3, as defined by the OSPF Flexible Algorithm Definition TLV in [I-D.ietf-lsr-flex-algo].

The following sub-sections define the sub-TLVs for the FAD TLV.

2.1. Flex Algo Exclude Any Affinity

The Flex Algo Exclude Any Affinity sub-TLV is an optional sub-TLV that is used to carry the affinity constraints [RFC2702] associated with the flex algo definition and enable the exclusion of links carrying any of the specified affinities from the computation of the specific algorithm as described in [I-D.ietf-lsr-flex-algo]. The affinity is expressed in terms of Extended Admin Group (EAG) as defined in [RFC7308].

The TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Type            |              Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Exclude-Any EAG (variable)                       //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- o Type: TBD (see IANA Considerations Section 3)

- o Length: variable, dependent on the size of the Extended Admin Group. MUST be a multiple of 4 octets.

- o Exclude-Any EAG : the bitmask used to represent the affinities to be excluded.

The information in the Flex Algo Exclude Any Affinity sub-TLV is derived from the IS-IS and OSPF protocol specific Flexible Algorithm Exclude Admin Group sub-TLV as defined in [I-D.ietf-lsr-flex-algo].

2.2. Flex Algo Include Any Affinity

The Flex Algo Incude Any Affinity sub-TLV is an optional sub-TLV that is used to carry the affinity constraints [RFC2702] associated with the flex algo definition and enable the inclusion of links carrying any of the specified affinities in the computation of the specific algorithm as described in [I-D.ietf-lsr-flex-algo]. The affinity is expressed in terms of Extended Admin Group (EAG) as defined in [RFC7308].

The TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Type            |              Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Include-Any EAG (variable)                        //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- o Type: TBD (see IANA Considerations Section 3)

- o Length: variable, dependent on the size of the Extended Admin Group. MUST be a multiple of 4 octets.

- o Include-Any EAG : the bitmask used to represent the affinities to be included.

The information in the Flex Algo Include Any Affinity sub-TLV is derived from the IS-IS and OSPF protocol specific Flexible Algorithm Include-Any Admin Group sub-TLV as defined in [I-D.ietf-lsr-flex-algo].

2.3. Flex Algo Include All Affinity

The Flex Algo Incude All Affinity sub-TLV is an optional sub-TLV that is used to carry the affinity constraints [RFC2702] associated with the flex algo definition and enable the inclusion of links carrying all of the specified affinities in the computation of the specific algorithm as described in [I-D.ietf-lsr-flex-algo]. The affinity is expressed in terms of Extended Admin Group (EAG) as defined in [RFC7308].

The TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Type            |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Include-All EAG (variable)                        //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- o Type: TBD (see IANA Considerations Section 3)

- o Length: variable, dependent on the size of the Extended Admin Group. MUST be a multiple of 4 octets.

- o Include-All EAG : the bitmask used to represent the affinities to be included.

The information in the Flex Algo Include All Affinity sub-TLV is derived from the IS-IS and OSPF protocol specific Flexible Algorithm Include-All Admin Group sub-TLV as defined in [I-D.ietf-lsr-flex-algo].

3. IANA Considerations

This document requests assigning code-points from the registry "BGP-LS Node Descriptor, Link Descriptor, Prefix Descriptor, and Attribute TLVs" based on table below. The column "IS-IS TLV/Sub-TLV" defined in the registry does not require any value and should be left empty.

Code Point	Description	Length
TBD	Flex Algorithm Definition TLV	variable
TBD	Flex Algo Exclude Any Affinity sub-TLV	variable
TBD	Flex Algo Include Any Affinity sub-TLV	variable
TBD	Flex Algo Include All Affinity sub-TLV	variable

4. Manageability Considerations

This section is structured as recommended in [RFC5706].

The new protocol extensions introduced in this document augment the existing IGP topology information that was distributed via [RFC7752]. Procedures and protocol extensions defined in this document do not affect the BGP protocol operations and management other than as discussed in the Manageability Considerations section of [RFC7752]. Specifically, the malformed NLRIs attribute tests in the Fault Management section of [RFC7752] now encompass the new TLVs for the BGP-LS NLRI in this document.

4.1. Operational Considerations

No additional operation considerations are defined in this document.

4.2. Management Considerations

No additional management considerations are defined in this document.

5. Security Considerations

The new protocol extensions introduced in this document augment the existing IGP topology information that was distributed via [RFC7752]. Procedures and protocol extensions defined in this document do not affect the BGP security model other than as discussed in the Security Considerations section of [RFC7752].

6. Acknowledgements

The authors would like to thank Les Ginsberg for his reviews and contributions to this work.

7. References

7.1. Normative References

[I-D.ietf-lsr-flex-algo]
Psenak, P., Hegde, S., Filsfils, C., Talaulikar, K., and A. Gulko, "IGP Flexible Algorithm", draft-ietf-lsr-flex-algo-01 (work in progress), November 2018.

[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>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

[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>.

7.2. Informative References

[I-D.ietf-idr-bgp-ls-segment-routing-ext]
Previdi, S., Talaulikar, K., Filsfils, C., Gredler, H., and M. Chen, "BGP Link-State extensions for Segment Routing", draft-ietf-idr-bgp-ls-segment-routing-ext-11 (work in progress), October 2018.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d., bogdanov@google.com, b., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-02 (work in progress), October 2018.

[RFC2702] Awduche, D., Malcolm, J., Agogbua, J., O'Dell, M., and J. McManus, "Requirements for Traffic Engineering Over MPLS", RFC 2702, DOI 10.17487/RFC2702, September 1999, <https://www.rfc-editor.org/info/rfc2702>.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <https://www.rfc-editor.org/info/rfc3209>.

[RFC5706] Harrington, D., "Guidelines for Considering Operations and Management of New Protocols and Protocol Extensions", RFC 5706, DOI 10.17487/RFC5706, November 2009, <https://www.rfc-editor.org/info/rfc5706>.

[RFC7308] Osborne, E., "Extended Administrative Groups in MPLS Traffic Engineering (MPLS-TE)", RFC 7308, DOI 10.17487/RFC7308, July 2014, <https://www.rfc-editor.org/info/rfc7308>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

Authors' Addresses

Ketan Talaulikar
Cisco Systems

Email: ketant@cisco.com

Peter Psenak
Cisco Systems
Slovakia

Email: ppsenak@cisco.com

Shawn Zandi
LinkedIn
USA

Email: szandi@linkedin.com

Gaurav Dawra
LinkedIn
USA

Email: gdawra.ietf@gmail.com

Inter-Domain Routing
Internet-Draft
Intended status: Standards Track
Expires: August 26, 2019

Z. Li
Huawei
S. Aldrin
Google, Inc
J. Tantsura
Apstra
G. Mirsky
ZTE Corp.
S. Zhuang
Huawei
K. Talaulikar
Cisco Systems
February 22, 2019

BGP Link-State Extensions for Seamless BFD
draft-li-idr-bgp-ls-sbfd-extensions-03

Abstract

Seamless Bidirectional Forwarding Detection (S-BFD) defines a simplified mechanism to use Bidirectional Forwarding Detection (BFD) with large portions of negotiation aspects eliminated, thus providing benefits such as quick provisioning as well as improved control and flexibility to network nodes initiating the path monitoring. The link-state routing protocols (IS-IS and OSPF) have been extended to advertise the Seamless BFD (S-BFD) Discriminators.

This draft defines extensions to the BGP Link-state address-family to carry the S-BFD Discriminators information via BGP.

Requirements Language

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.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Seamless Bidirectional Forwarding Detection (S-BFD) [RFC7880] defines a simplified mechanism to use Bidirectional Forwarding Detection (BFD) [RFC5880] with large portions of negotiation aspects eliminated, thus providing benefits such as quick provisioning as well as improved control and flexibility to network nodes initiating the path monitoring.

For monitoring of a service path end-to-end via S-BFD, the headend/ initiator node needs to know the S-BFD Discriminator of the destination/tail-end node of that service. The link-state routing protocols (IS-IS, OSPF and OSPFv3) have been extended to advertise the S-BFD Discriminators. With this a initiator node can learn the S-BFD discriminator for all nodes within its IGP area/level or optionally within the domain. With networks being divided into multiple IGP domains for scaling and operational considerations, the service endpoints that require end to end S-BFD monitoring often span across IGP domains.

BGP Link-State (BGP-LS) [RFC7752] enables the collection and distribution of IGP link-state topology information via BGP sessions across IGP areas/levels and domains. The S-BFD discriminator(s) of a node can thus be distributed along with the topology information via BGP-LS across IGP domains and even across multiple Autonomous Systems (AS) within an administrative domain.

This draft defines extensions to BGP-LS for carrying the S-BFD Discriminators information.

2. Terminology

This memo makes use of the terms defined in [RFC7880].

3. Problem and Requirement

Seamless MPLS [I-D.ietf-mpls-seamless-mpls] extends the core domain and integrates aggregation and access domains into a single MPLS domain. In a large network, the core and aggregation networks can be organized as different ASes. Although the core and aggregation networks are segmented into different ASes, an E2E LSP can be created using hierarchical BGP signaled LSPs based on iBGP labeled unicast within each AS, and eBGP labeled unicast to extend the LSP across AS boundaries. This provides a seamless MPLS transport connectivity for any two service end-points across the entire domain. In order to detect failures for such end to end services and trigger faster protection and/or re-routing, S-BFD MAY be used for the Service Layer (e.g. for MPLS VPNs, PW, etc.) or the Transport Layer monitoring. This brings up the need for setting up S-BFD session spanning across AS domains.

In a similar Segment Routing (SR) [RFC8402] multi-domain network, an end to end SR Policy [I-D.ietf-spring-segment-routing-policy] path may be provisioned between service end-points across domains either via local provisioning or by a controller or signalled from a Path Computation Engine (PCE). Monitoring using S-BFD can similarly be setup for such a SR Policy.

Extending the automatic discovery of S-BFD discriminators of nodes from within the IGP domain to across the administrative domain using BGP-LS enables setting up of S-BFD sessions on demand across IGP domains. The S-BFD discriminators for service end point nodes MAY be learnt by the PCE or a controller via the BGP-LS feed that it gets from across IGP domains and it can signal or provision the remote S-BFD discriminator on the initiator node on demand when S-BFD monitoring is required. The mechanisms for the signaling of the S-BFD discriminator from the PCE/controller to the initiator node and setup of the S-BFD session is outside the scope of this document.

Additionally, the service end-points themselves MAY also learn the S-BFD discriminator of the remote nodes themselves by receiving the BGP-LS feed via a route reflector (RR) or a centralized BGP Speaker that is consolidating the topology information across the domains. The initiator node can then itself setup the S-BFD session to the remote node without a controller/PCE assistance.

While this document takes examples of MPLS and SR paths, the S-BFD discriminator advertisement mechanism is applicable for any S-BFD use-case in general.

4. BGP-LS Extensions for S-BFD Discriminator

The BGP-LS [RFC7752] specifies the Node NLRI for advertisement of nodes and their attributes using the BGP-LS Attribute. The S-BFD discriminators of a node are considered as its node level attribute and advertised as such.

This document defines a new BGP-LS Attribute TLV called the S-BFD Discriminators TLV and its format is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Discriminator 1                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Discriminator 2 (Optional)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Discriminator n (Optional)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: S-BFD Discriminators TLV

where:

- o Type: TBD (see IANA Considerations Section 5)

- o Length: variable. Minimum of 8 octets and increments of 4 octets there on for each additional discriminator

- o Discriminators : multiples of 4 octets, each carrying a S-BFD local discriminator value of the node. At least one discriminator MUST be included in the TLV.

The S-BFD Discriminators TLV can only be added to the BGP-LS Attribute associated with the Node NLRI that originates the corresponding underlying IGP TLV/sub-TLV as described below. This information is derived from the protocol specific advertisements as below..

- o IS-IS, as defined by the S-BFD Discriminators sub-TLV in [RFC7883].

- o OSPFv2/OSPFv3, as defined by the S-BFD Discriminators TLV in [RFC7884].

When the node is not running any of the IGPs but running a protocol like BGP, then the locally provisioned S-BFD discriminators of the node MAY be originated as part of the BGP-LS attribute within the Node NLRI corresponding to the local node.

5. IANA Considerations

This document requests assigning code-points from the registry "BGP-LS Node Descriptor, Link Descriptor, Prefix Descriptor, and Attribute TLVs" based on table below. The column "IS-IS TLV/Sub-TLV" defined in the registry does not require any value and should be left empty.

Code Point	Description	Length
TBD	S-BFD Discriminators TLV	variable

6. Manageability Considerations

This section is structured as recommended in [RFC5706].

The new protocol extensions introduced in this document augment the existing IGP topology information that was distributed via [RFC7752]. Procedures and protocol extensions defined in this document do not affect the BGP protocol operations and management other than as discussed in the Manageability Considerations section of [RFC7752]. Specifically, the malformed NLRIs attribute tests in the Fault Management section of [RFC7752] now encompass the new TLVs for the BGP-LS NLRI in this document.

6.1. Operational Considerations

No additional operation considerations are defined in this document.

6.2. Management Considerations

No additional management considerations are defined in this document.

7. Security Considerations

The new protocol extensions introduced in this document augment the existing IGP topology information that was distributed via [RFC7752]. Procedures and protocol extensions defined in this document do not affect the BGP security model other than as discussed in the Security Considerations section of [RFC7752]. More specifically the aspects related to limiting the nodes and consumers with which the topology information is shared via BGP-LS to trusted entities within an administrative domain.

Advertising the S-BFD Discriminators via BGP-LS makes it possible for attackers to initiate S-BFD sessions using the advertised

information. The vulnerabilities this poses and how to mitigate them are discussed in [RFC7752].

8. Acknowledgements

The authors would like to thank Nan Wu for his contributions to this work.

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>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

[RFC7880] Pignataro, C., Ward, D., Akiya, N., Bhatia, M., and S. Pallagatti, "Seamless Bidirectional Forwarding Detection (S-BFD)", RFC 7880, DOI 10.17487/RFC7880, July 2016, <https://www.rfc-editor.org/info/rfc7880>.

[RFC7883] Ginsberg, L., Akiya, N., and M. Chen, "Advertising Seamless Bidirectional Forwarding Detection (S-BFD) Discriminators in IS-IS", RFC 7883, DOI 10.17487/RFC7883, July 2016, <https://www.rfc-editor.org/info/rfc7883>.

[RFC7884] Pignataro, C., Bhatia, M., Aldrin, S., and T. Ranganath, "OSPF Extensions to Advertise Seamless Bidirectional Forwarding Detection (S-BFD) Target Discriminators", RFC 7884, DOI 10.17487/RFC7884, July 2016, <https://www.rfc-editor.org/info/rfc7884>.

[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>.

9.2. Informative References

[I-D.ietf-mpls-seamless-mpls]
Leymann, N., Decraene, B., Filsfils, C., Konstantynowicz, M., and D. Steinberg, "Seamless MPLS Architecture", draft-ietf-mpls-seamless-mpls-07 (work in progress), June 2014.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d., bogdanov@google.com, b., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-02 (work in progress), October 2018.

[RFC5706] Harrington, D., "Guidelines for Considering Operations and Management of New Protocols and Protocol Extensions", RFC 5706, DOI 10.17487/RFC5706, November 2009, <https://www.rfc-editor.org/info/rfc5706>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

Authors' Addresses

Zhenbin Li
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com

Sam Aldrin
Google, Inc

Email: aldrin.ietf@gmail.com

Jeff Tantsura
Apstra

Email: jefftant.ietf@gmail.com

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Shunwan Zhuang
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: zhuangshunwan@huawei.com

Ketan Talaulikar
Cisco Systems
India

Email: ketant@cisco.com

Interdomain Routing Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 9, 2020

C. Li
Huawei Technologies
H. Chen
China Telecom
M. Chen
J. Dong
Z. Li
Huawei Technologies
August 8, 2019

SR Policies Extensions for Path Segment and Bidirectional Path in BGP-LS
draft-li-idr-bgp-ls-sr-policy-path-segment-03

Abstract

This document specifies the way of collecting configuration and states of SR policies carrying Path Segment and bidirectional path information by using BPG-LS. Such information can be used by external conponents for many use cases such as performance measurement, path re-optimization and end-to-end protection.

Requirements Language

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

Status of This Memo

Table of Contents

1. Introduction

Segment routing (SR) [RFC8402] is a source routing paradigm that allows the ingress node steers packets into a specific path according to the Segment Routing Policy [I-D.ietf-spring-segment-routing-policy].

However, the SR Policies defined in [I-D.ietf-spring-segment-routing-policy] only supports unidirectional SR paths and there is no path ID in a Segment List to identify an SR path. For identifying an SR path and supporting bidirectional path [I-D.ietf-spring-mpls-path-segment], new policies carrying Path Segment and bidirectional path information are defined in

[I-D.li-idr-sr-policy-path-segment-distribution], as well as the extensions to BGP to distribute new SR policies. The Path Segment can be a Path Segment in SR-MPLS [I-D.ietf-spring-mpls-path-segment] , or other IDs that can identify a path.

In many network scenarios, the configuration and state of each TE Policy is required by a controller which allows the network operator to optimize several functions and operations through the use of a controller aware of both topology and state information [I-D.ietf-idr-te-lsp-distribution].

To collect the TE Policy information that is locally available in a router, [I-D.ietf-idr-te-lsp-distribution] describes a new mechanism by using BGP-LS update messages.

Based on the mechanism defined in [I-D.ietf-idr-te-lsp-distribution], this document describes a mechanism to distribute configuration and states of the new SR policies defined in [I-D.li-idr-sr-policy-path-segment-distribution] to external components using BGP-LS.

2. Terminology

This memo makes use of the terms defined in [RFC8402] and [I-D.ietf-idr-te-lsp-distribution].

3. Carrying SR Path Sub-TLVs in BGP-LS

A mechanism to collect states of SR Policies via BGP-LS is proposed by [I-D.ietf-idr-te-lsp-distribution]. The characteristics of an SR policy can be described by a TE Policy State TLV, which is carried in the optional non-transitive BGP Attribute "LINK_STATE Attribute" defined in [RFC7752]. The TE Policy State TLV contains several sub-TLVs such as SR TE Policy sub-TLVs. Rather than replicating SR TE Policy sub-TLVs, [I-D.ietf-idr-te-lsp-distribution] reuses the equivalent sub-TLVs as defined in [I-D.ietf-idr-segment-routing-te-policy].

[I-D.li-idr-sr-policy-path-segment-distribution] defines the BGP extensions for Path Segment. The Path Segment can appear at both segment-list level and candidate path level upon the use case. The encoding is shown below.

```
SR Policy SAFI NLRI: <Distinguisher, Policy-Color, Endpoint>
Attributes:
   Tunnel Encaps Attribute (23)
      Tunnel Type: SR Policy
          Binding SID
          Preference
          Priority
          Policy Name
          Explicit NULL Label Policy (ENLP)
          Path Segment
          Segment List
              Weight
              Path Segment
              Segment
              Segment
              ...
          Segment List
              Weight
              Path Segment
              Segment
              Segment
              ...
          ...
```

Figure 1. Path Segment in SR policy

Also, [I-D.li-idr-sr-policy-path-segment-distribution] defines SR policy extensions for bidirectional SR path, the encoding is shown below:

```
SR Policy SAFI NLRI: <Distinguisher, Policy-Color, Endpoint>
    Attributes: Tunnel Encaps Attribute (23)
    Tunnel Type: SR Policy
        Binding SID
        Preference
        Priority
        Policy Name
        Explicit NULL Label Policy (ENLP)
        Bidirectioanl Path
            Segment List
                Weight
                Path Segment
                Segment
                Segment
                ...
            Reverse Segment List
                Weight
                Path Segment
                Segment
                Segment
                ...
```

Figure 2. SR policy for Bidirectional path

In order to collect configuration and states of unidirectional and bidirectional SR policies defined in [I-D.li-idr-sr-policy-path-segment-distribution], new sub-TLVs in SR TE Policy sub-TLVs should be defined. Likewise, rather than replicating SR Policy sub-TLVs, this document can reuse the equivalent sub-TLVs as defined in [I-D.li-idr-sr-policy-path-segment-distribution].

3.1. SR Path Segment Sub-TLV

This section reuses the SR Path Segment sub-TLV defined in [I-D.li-idr-sr-policy-path-segment-distribution] to describe a Path Segment , and it can be included in the Segment List sub-TLV as defined in [I-D.ietf-idr-te-lsp-distribution] . An SR Path Segment sub-TLV can be associated with an SR path specified by a Segment List sub-TLV, and it MUST appear only once within a Segment List sub-TLV. Also, it can be used for identifying an SR candidate path or an SR Policy defined in [I-D.ietf-spring-segment-routing-policy].

The format of Path Segment TLV is included below for reference.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |    Length     |    Flag       |      ST       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Path Segment (Variable)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2. Path Segment sub-TLV

All fields, including type and length, are defined in [I-D.li-idr-sr-policy-path-segment-distribution].

3.2. Sub-TLVs for Bidirectional Path

In some scenarios like mobile backhaul transport network, there are requirements to support bidirectional path. In SR, a bidirectional path can be represented as a binding of two unidirectional SR paths [I-D.ietf-spring-mpls-path-segment]. [I-D.li-idr-sr-policy-path-segment-distribution] defines new sub-TLVs to describe an SR bidirectional path. An SR policy carrying SR bidirectional path information is expressed in Figure 1.

3.2.1. SR Bidirectional Path Sub-TLV

This section reuses the SR bidirectional path sub-TLV defined in [I-D.li-idr-sr-policy-path-segment-distribution] to specify a bidirectional path, which contains a Segment List sub-TLV [I-D.ietf-idr-segment-routing-te-policy] and an associated Reverse Path Segment List as defined in [I-D.li-idr-sr-policy-path-segment-distribution]. The SR bidirectional path sub-TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type       |             Length            |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sub-TLVs (Variable)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3. SR Bidirectional path sub-TLV

All fields, including type and length, are defined in [I-D.li-idr-sr-policy-path-segment-distribution].

3.2.2. SR Reverse Path Segment List Sub-TLV

This section reuses the SR Reverse Path Segment List sub-TLV defined in [I-D.li-idr-sr-policy-path-segment-distribution] to specify an reverse SR path associated with the path specified by the Segment List in the same SR Bidirectional Path Sub-TLV, and it has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type       |             Length            |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sub-TLVs (Variable)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4. SR Reverse Path Segment List Sub-TLV

All fields, including type and length, are defined in [I-D.li-idr-sr-policy-path-segment-distribution].

4. Operations

No new operation procedures are defined in this document, the operations procedures of [RFC7752] can apply to this document.

Typically but not limited to, the uni/bidirectional SR policies carrying path identification information can be distributed by the ingress node.

Generally, BGP-LS is used for collecting link states and synchronizing with the external component. The consumer of the uni/bidirectional SR policies carrying path identification information is not BGP LS process by itself, and it can be any applications such as performance measurement [I-D.gandhi-spring-udp-pm] and path re-coputation or re-optimization, etc. The operation of sending information to other precesses is out of scope of this document.

5. IANA Considerations

5.1. BGP-LS TLVs

IANA maintains a registry called "Border Gateway Protocol - Link State (BGP-LS) Parameters" with a sub-registry called "Node Anchor, Link Descriptor and Link Attribute TLVs". The following TLV codepoints are suggested (for early allocation by IANA):

Codepoint	Description	Reference
1212	Path Segment sub-TLV	This document
1213	SR Bidirectional Path sub-TLV	This document
1214	Reverse Segment List sub-TLV	This document

5.2. BGP-LS SR Segment Descriptors

This document defines new sub-TLVs in the registry "SR Segment Descriptor Types" [I-D.ietf-idr-te-lsp-distribution] to be assigned by IANA:

Codepoint	Description	Reference
14	Path Segment sub-TLV	This document

6. Security Considerations

TBA

7. Acknowledgements

Many thanks to Shraddha Hedge for her detailed review and professional comments.

8. References

8.1. Normative References

[I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Mattes, P., Rosen, E., Jain, D., and S. Lin, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-07 (work in progress), July 2019.

[I-D.ietf-idr-te-lsp-distribution]
Previdi, S., Talaulikar, K., Dong, J., Chen, M., Gredler, H., and J. Tantsura, "Distribution of Traffic Engineering (TE) Policies and State using BGP-LS", draft-ietf-idr-te-lsp-distribution-11 (work in progress), May 2019.

[I-D.ietf-spring-mpls-path-segment]
Cheng, W., Li, H., Chen, M., Gandhi, R., and R. Zigler, "Path Segment in MPLS Based Segment Routing Network", draft-ietf-spring-mpls-path-segment-00 (work in progress), March 2019.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d., bogdanov@google.com, b., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-03 (work in progress), May 2019.

[I-D.li-idr-sr-policy-path-segment-distribution]
Li, C., Chen, M., Dong, J., and Z. Li, "Segment Routing Policies for Path Segment and Bidirectional Path", draft-li-idr-sr-policy-path-segment-distribution-01 (work in progress), October 2018.

[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>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

8.2. Informative References

[I-D.gandhi-spring-udp-pm]
Gandhi, R., Filsfils, C., daniel.voyer@bell.ca, d., Salsano, S., Ventre, P., and M. Chen, "UDP Path for In-band Performance Measurement for Segment Routing Networks", draft-gandhi-spring-udp-pm-02 (work in progress), September 2018.

[I-D.ietf-mpls-bfd-directed]
Mirsky, G., Tantsura, J., Varlashkin, I., and M. Chen, "Bidirectional Forwarding Detection (BFD) Directed Return Path", draft-ietf-mpls-bfd-directed-11 (work in progress), April 2019.

Authors' Addresses

Cheng Li
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: chengli13@huawei.com

Huanan Chen
China Telecom
109 West Zhongshan Ave
Guangzhou
China

Email: chenhn8.gd@chinatelecom.cn

Mach(Guoyi) Chen
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: Mach.chen@huawei.com

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: jie.dong@huawei.com

Zhenbin Li
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com

Interdomain Routing Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

C. Li
M. Chen
J. Dong
Z. Li
Huawei Technologies

October 22, 2018

Segment Routing Policies for Path Segment and Bidirectional Path
draft-li-idr-sr-policy-path-segment-distribution-01

Abstract

An SR policy is a set of candidate SR paths consisting of one or more segment lists with necessary path attributes. For each SR path, it may also have its own path attributes, and Path Segment is one of them. A Path Segment is defined to identify an SR path, which can be used for performance measurement, path correlation, and end-2-end path protection. Path Segment can be also used to correlate two unidirctional SR paths into a bidirectional SR path which is required in some scenarios, for example, mobile backhaul transport network.

This document defines extensions to BGP to distribute SR policies carrying Path Segment and bidirectional path information.

Requirements Language

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

Status of This Memo

Table of Contents

1. Introduction

Segment routing (SR) [RFC8402] is a source routing paradigm that explicitly indicates the forwarding path for packets at the ingress node. The ingress node steers packets into a specific path according to the Segment Routing Policy (SR Policy) as defined in [I-D.ietf-spring-segment-routing-policy]. For distributing SR policies to the headend, [I-D.ietf-idr-segment-routing-te-policy] specifies a mechanism by using BGP, and new sub-TLVs are defined for SR Policies in BGP UPDATE message.

In many use cases such as performance measurement, the path to which the packets belong is required to be identified. Futhermore, in some scenarios, for example, mobile backhaul transport network, there are requirements to support bidirectional path. However, there is no

path identification information for each Segment List in the SR Policies defined in [I-D.ietf-spring-segment-routing-policy]. Also, the SR Policies defined in [I-D.ietf-spring-segment-routing-policy] only supports unidirectional SR paths.

Therefore, this document defines the extension to SR policies that carry Path Segment in the Segment List and support bidirectional path. The Path Segment can be a Path Segment in SR-MPLS [I-D.cheng-spring-mpls-path-segment] , or a Path Segment in SRv6 [I-D.li-spring-srv6-path-segment], or other IDs that can identify a path. Also, this document defines extensions to BGP to distribute SR policies carriying Path Segment and bidirectional path information.

2. Terminology

This memo makes use of the terms defined in [RFC8402] and [I-D.ietf-idr-segment-routing-te-policy].

3. SR Policy for Path Identifier

As defined in [I-D.ietf-idr-segment-routing-te-policy] , the SR Policy encoding structure is as follows:

```
SR Policy SAFI NLRI: <Distinguisher, Policy-Color, Endpoint>
Attributes:
   Tunnel Encaps Attribute (23)
      Tunnel Type: SR Policy
          Binding SID
          Preference
          Priority
          Policy Name
          Explicit NULL Label Policy (ENLP)
          Segment List
              Weight
              Segment
              Segment
              ...
          ...
```

An SR path can be specified by an Segment List sub-TLV that contains a set of segment sub-TLVs and other sub-TLVs as shown above. As defined in [I-D.ietf-spring-segment-routing-policy], a candidate path includes multiple SR paths specified by SID list. The Path Segment can be used for idendifying an SR path(specified by SID list). Also, it can be used for identifying an SR candidate path or an SR Policy in some use cases if needed. New SR Policy encoding structure is expressed as below:

```
SR Policy SAFI NLRI: <Distinguisher, Policy-Color, Endpoint>
Attributes:
   Tunnel Encaps Attribute (23)
      Tunnel Type: SR Policy
          Binding SID
          Preference
          Priority
          Policy Name
          Explicit NULL Label Policy (ENLP)
          Path Segment
          Segment List
              Weight
              Path Segment
              Segment
              Segment
              ...
          Segment List
              Weight
              Path Segment
              Segment
              Segment
              ...
          ...
```

3.1. SR Path Segment Sub-TLV

This section defines an SR Path Segment sub-TLV.

An SR Path Segment sub-TLV can be included in the segment list sub-TLV to identify an SID list, and it MUST appear only once within a Segment List sub-TLV. It has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |    Length     |    Flag       |      ST       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Path Segment (Variable)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1. Path Segment sub-TLV

Where:

Type: to be assigned by IANA (suggested value 10).

Length: the total length of the value field not including Type and Length fields.

Flag: 8 bits of flags. Following flags are defined:

```
  0  1  2  3  4  5  6  7
+--+--+--+--+--+--+--+--+
|    Reserved        |G |
+--+--+--+--+--+--+--+--+
```

G-Flag: Global flag. Set when the Path Segment is global within an SR domain.

Reserved: 5 bits reserved and MUST be set to 0 on transmission and MUST be ignored on receipt.

ST: Segment type, specifies the type of the Path Segment, and it has following types:

- o 0: SR-MPLS Path Segment
- o 1: SRv6 Path Segment
- o 2-255:Reserved

Path Segment: The Path Segment of an SR path. The Path Segment type is indicated by the Segment Type(ST) field. It can be a Path Segment in SR-MPLS [I-D.cheng-spring-mpls-path-segment], or a Path Segment in SRv6 [I-D.li-spring-srv6-path-segment], or other IDs that can identify a path.

4. SR Policy for Bidirectional Path

In some scenarios, for example, mobile backhaul transport network, there are requirements to support bidirectional path. In SR, a bidirectional path can be represented as a binding of two unidirectional SR paths. This document also defines new sub-TLVs to describe an SR bidirectional path. An SR policy carrying SR bidirectional path information is expressed as below:

```
SR Policy SAFI NLRI: <Distinguisher, Policy-Color, Endpoint>
    Attributes: Tunnel Encaps Attribute (23)
    Tunnel Type: SR Policy
        Binding SID
        Preference
        Priority
        Policy Name
        Explicit NULL Label Policy (ENLP)
        Bidirectioanl Path
            Segment List
                Weight
                Path Segment
                Segment
                Segment
                ...
            Reverse Segment List
                Weight
                Path Segment
                Segment
                Segment
                ...
```

4.1. SR Bidirectional Path Sub-TLV

This section defines an SR bidirectional path sub-TLV to specify a bidirectional path, which contains a Segment List sub-TLV [I-D.ietf-idr-segment-routing-te-policy] and an associated Reverse Path Segment List as defined at section 4.2. The SR bidirectional path sub-TLV has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type       |             Length            |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sub-TLVs (Variable)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2. SR Bidirectional path sub-TLV

Where:

Type: TBA, and the suggest value is 14.

Length: the total length of the sub-TLVs encoded within the SR Bidirectional Path Sub-TLV not including Type and Length fields.

RESERVED: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt.

Sub-TLVs:

- o An Segment List sub-TLV
- o An associated Reverse Path Segment List sub-TLV

4.2. SR Reverse Path Segment List Sub-TLV

An SR Reverse Path Segment List sub-TLV is defined to specify an SR reverse path associated with the path specified by the Segment List in the same SR Bidirectional Path Sub-TLV, and it has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Type       |             Length            |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sub-TLVs (Variable)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2. SR Reverse Path Segment List Sub-TLV

where:

Type: TBA, and suggest value is 127.

Length: the total length of the sub-TLVs encoded within the SR Reverse Path Segment List Sub-TLV not including the Type and Length fields.

RESERVED: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt.

sub-TLVs, reuse the sub-TLVs in Segment List defined in [I-D.ietf-idr-segment-routing-te-policy].

- o An optional single Weight sub-TLV.
- o An mandatory SR Path Segment sub-TLV that contains the Path Segment of the reverse SR path.
- o Zero or more Segment sub-TLVs to specify the reverse SR path.

The Segment sub-TLVs in the Reverse Path Segment List sub-TLV provides the information of the reverse SR path, which can be used

for directing egress BFD peer to use specific path for the reverse direction of the BFD session [I-D.ietf-mpls-bfd-directed] or other applications.

5. Operations

The document does not bring new operation beyong the description of operations defined in [I-D.ietf-idr-segment-routing-te-policy]. The existing operations defined in [I-D.ietf-idr-segment-routing-te-policy] can apply to this document directly.

Typically but not limit to, the unidirectional or bidirectional SR policies carrying path identification infomation are configured by a controller.

After configuration, the unidirectional or bidirectional SR policies carrying path identification infomation will be advertised by BGP update messages. The operation of advertisement is the same as defined in [I-D.ietf-idr-segment-routing-te-policy], as well as the receiption.

The consumer of the unidirectional or bidirectional SR policies is not the BGP process, it can be any applications, such as performance measurement [I-D.gandhi-spring-udp-pm]. The operation of sending information to consumers is out of scope of this document.

6. IANA Considerations

TBA

7. Security Considerations

TBA

8. Acknowledgements

TBA

9. References

9.1. Normative References

[I-D.cheng-spring-mpls-path-segment]
Cheng, W., Wang, L., Li, H., Chen, M., Gandhi, R., Zigler, R., and S. Zhan, "Path Segment in MPLS Based Segment Routing Network", draft-cheng-spring-mpls-path-segment-03 (work in progress), October 2018.

[I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Jain, D., Mattes, P., Rosen, E., and S. Lin, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-04 (work in progress), July 2018.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d., bogdanov@google.com, b., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-01 (work in progress), June 2018.

[I-D.li-spring-srv6-path-segment]
Li, C., Chen, M., Dhody, D., Li, Z., Dong, J., and R. Gandhi, "Path Segment for SRv6 (Segment Routing in IPv6)", draft-li-spring-srv6-path-segment-00 (work in progress), October 2018.

[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>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

9.2. Informative References

[I-D.gandhi-spring-udp-pm]
Gandhi, R., Filsfils, C., daniel.voyer@bell.ca, d., Salsano, S., Ventre, P., and M. Chen, "UDP Path for In-band Performance Measurement for Segment Routing Networks", draft-gandhi-spring-udp-pm-02 (work in progress), September 2018.

[I-D.ietf-mpls-bfd-directed]
Mirsky, G., Tantsura, J., Varlashkin, I., and M. Chen, "Bidirectional Forwarding Detection (BFD) Directed Return Path", draft-ietf-mpls-bfd-directed-10 (work in progress), September 2018.

Authors' Addresses

Cheng Li
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: chengli13@huawei.com

Mach(Guoyi) Chen
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: Mach.chen@huawei.com

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: jie.dong@huawei.com

Zhenbin Li
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com

IDR Working Group
Internet-Draft
Intended status: Informational
Expires: January 3, 2019

K. Sriram, Ed.
USA NIST

July 2, 2018

Design Discussion of Route Leaks Solution Methods
draft-sriram-idr-route-leak-solution-discussion-00

Abstract

This document captures the design rationale of the route leaks solution document [draft-ietf-idr-route-leak-detection-mitigation]. The designers needed to balance many competing factors, and this document provides insights into the design questions and their resolution.

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 January 3, 2019.

Table of Contents

1. Introduction

This document captures the design rationale of the route leaks solution document [I-D.ietf-idr-route-leak-detection-mitigation]. The designers needed to balance many competing factors, and this document provides insights into the design questions and their resolution.

2. Related Prior Work

The solution described in [I-D.ietf-idr-route-leak-detection-mitigation] is based on setting an attribute in BGP route announcement to manage the transmission/ receipt of the announcement based on the type of neighbor (e.g., customer, transit provider, etc.). Documented prior work related to this basic idea and mechanism dates back to at least the 1980’s. Some examples of prior work are: (1) Information flow rules described in [proceedings-sixth-ietf] (see pp. 195-196); (2) Link Type described in [RFC1105-obsolete] (see pp. 4-5); (3) Hierarchical Recording described in [draft-kunzinger-idrp-ISO10747-01] (see Section 6.3.1.12). The problem of route leaks and possible solution mechanisms based on encoding peering-link type information, e.g., P2C

(i.e., Transit-Provider to Customer), C2P (i.e., Customer to Transit-Provider), p2p (i.e., peer to peer) etc., in BGPsec updates and protecting the same under BGPsec path signatures have been discussed in IETF SIDR WG at least since 2011. [draft-dickson-sidr-route-leak-solns] attempted to describe these mechanisms in a BGPsec context. The draft expired in 2012. [draft-dickson-sidr-route-leak-solns] defined neighbor relationships on a per link basis, but in [I-D.ietf-idr-route-leak-detection-mitigation] the relationship is encoded per prefix, as routes for prefixes with different peering relationships may be sent over the same link. Also [draft-dickson-sidr-route-leak-solns] proposed a second signature block for the link type encoding, separate from the path signature block in BGPsec. By contrast, in [I-D.ietf-idr-route-leak-detection-mitigation], when BGPsec-based solution is considered, cryptographic protection is provided for Route-Leak Protection (RLP) encoding using the same signature block as that for path signatures (see Section 3.2.2 in [I-D.ietf-idr-route-leak-detection-mitigation]).

3. Design Rationale and Discussion

This section provides design justifications for the solution specified in [I-D.ietf-idr-route-leak-detection-mitigation], and also answers some questions that are anticipated or have been raised in the IETF IDR and SIDR working group meetings.

3.1. Explanation of Rules 1 and 2 in the solution document

In Section 3.3 in [I-D.ietf-idr-route-leak-detection-mitigation], Rules 1 and 2 were stated and the route leak mitigation policy was based on these rules to preserve the property of stable route convergence (i.e., avoid possibility of persistent route oscillations). Rule 1 is stated as follows:

- Rule 1: If ISP A receives a route r1 from customer AS C and another route r2 from provider (or peer) AS B (for the same prefix), and both routes r1 and r2 contain AS C and AS X (any X not equal to C) in the path and contain [X] in their RLP Attributes, then prioritize the customer (AS C) route over the provider (or peer) route.

The rationale for Rule 1 can be developed as follows.

Preference condition for route stability: Prefer customer routes over peer or provider routes (see pp. 25-27 in [Gao-Rexford]).

Topology condition for route stability: No cycle of customer-provider relationships (see pp. 25-27 in [Gao-Rexford]).

Route-Leak Detection Theorem: Let it be given that ISP A receives a route r1 from customer AS C and another route r2 from provider AS B (for the same prefix), and each of the routes r1 and r2 contains AS C and AS X in its AS path and contains [X] in its RLP Attribute. Then, clearly r1 is in violation of [X]. It follows that r2 is also necessarily in violation of [X].

Proof: Let us suppose that r2 is not in violation of [X]. That implies that r2's path from C to B to A included only P2C links. That would mean that there is a cycle of customer-provider relationships involving the ASes in the AS path in r2. However, any such cycle is ruled out in practice by the topology condition for route stability as stated above. QED.

Corollary 1: The route leak detection theorem holds also when "provider AS B" in the theorem is replaced by "peer AS B". (Here peer means a lateral peer.)

Proof: Since r2 contains [X] in the RLP Attribute set by an AS prior to peer AS B, it follows that r2 is in violation of [X]. QED.

It can be observed that Rule 1 follows from the combination of the Theorem, Corollary 1 and the preference condition for route stability (stated above).

In Section 3.3 in [I-D.ietf-idr-route-leak-detection-mitigation], Rule 2 is stated as follows:

- Rule 2: If ISP A receives a route r1 from peer AS C and another route r2 from provider AS B (for the same prefix), and both routes r1 and r2 contain AS C and AS X (any X not equal to C) in the path and contain [X] in their RLP Attributes, then prioritize the peer (AS C) route over the provider (AS B) route.

The rationale for Rule 2 can be developed as follows.

Corollary 2: The route leak detection theorem holds also when "customer AS C" in the theorem is replaced by "peer AS C".

Proof for Corollary 2: Let us suppose that r2 is not in violation of [X]. That implies that r2's path from C to B to A included only P2C links. This results in a topology in which A's lateral peer B is also A's transit provider's transit provider. This gives rise to possibility of looping of routes since A can send routes to its transit B, B can forward the routes to its transit C, and C can

forward the routes to its peer A. But such looping is forbidden by the topology condition stated above.

It can be observed that Rule 2 follows from Corollary 2. In essence, if the provider route (r2) is a detoured (longer) version of the lateral peer route (r1), and violates the same RLP [X] as does the peer route, then prefer the shorter route (r2) via the peer.

Rules 1 and 2 are

3.2. Is route-leak solution without cryptographic protection an attack vector?

It has been asked if a route-leak solution without BGPsec, i.e., when RLP Fields are not protected, can turn into a new attack vector. The answer seems to be: not really! Even the NLRI and AS_PATH in BGP updates are attack vectors, and RPKI/OV/BGPsec seek to fix that. Consider the following. Say, if 99% of route leaks are accidental and 1% are malicious, and if route-leak solution without BGPsec eliminates the 99%, then perhaps it is worth it (step in the right direction). When BGPsec comes into deployment, the route-leak protection (RLP) bits can be mapped into BGPsec (using the Flags field) and then necessary security will be in place as well (within each BGPsec island as and when they emerge).

Further, let us consider the worst-case damage that can be caused by maliciously manipulating the RLP Field values in an implementation without cryptographic protection (i.e., sans BGPsec). Manipulation of the RLP bits can result in one of two types of attacks: (a) Upgrade attack and (b) Downgrade attack. Descriptions and discussions about these attacks follow. In what follows, P2C stands for transit provider to customer (Down); C2P stands for customer to transit provider (Up), and p2p stands for peer to peer (lateral or non-transit relationship).

(a) Upgrade attack: An AS that wants to intentionally leak a route would alter the RLP encodings for the preceding hops from 1 (i.e., ’Do not Propagate Up or Lateral’) to 0 (default) wherever applicable. This poses no problem for a route that keeps propagating in the ’Down’ (P2C) direction. However, for a route that propagates ’Up’ (C2P) or ’Lateral’ (p2p), the worst that can happen is that a route leak goes undetected. That is, a receiving router would not be able to detect the leak for the route in question by the RLP mechanism described here. However, the receiving router may still detect and mitigate it in some cases by applying other means such as prefix filters [RFC7454] [NIST-800-54]. If some malicious leaks go undetected (when RLP is deployed without BGPsec) that is possibly a

small price to pay for the ability to detect the bulk of route leaks that are accidental.

(b) Downgrade attack: RLP encoding is set to 1 (i.e., 'Do not Propagate Up or Lateral') when it should be set to 0 (default). This would result in a route being mis-detected and marked as a route leak. By default, RLP encoding is set to 0, and that helps reduce errors of this kind (i.e., accidental downgrade incidents). Every AS or ISP wants reachability for prefixes it originates and for its customer prefixes. So, an AS or ISP is not likely to change an RLP value 0 to 1 intentionally. If a route leak is detected (due to intentional or accidental downgrade) by a receiving router, it would prefer an alternate 'clean' route from a transit provider or peer over a 'marked' route from a customer. It may end up with a suboptimal path. In order to have reachability, the receiving router would accept a 'marked' route if there is no alternative that is 'clean'. So, RLP downgrade attacks (intentional or accidental) would be quite rare, and the consequences do not appear to be grave.

3.3. Combining results of route-leak detection, OV and BGPsec validation for path selection decision

Combining the results of route-leak detection, OV, and BGPsec validation for path selection decision is up to local policy in a receiving router. As an example, a router may always give precedence to outcomes of OV and BGPsec validation over that of route-leak detection. That is, if an update fails OV or BGPsec validation, then the update is not considered a candidate for path selection. Instead, an alternate update is chosen that passed OV and BGPsec validation and additionally was not marked as route leak.

If only OV is deployed (and not BGPsec), then there are six possible combinations between OV and route-leak detection outcomes. Because there are three possible outcomes for OV (NotFound, Valid, and Invalid) and two possible outcomes for route-leak detection (marked as leak and not marked). If OV and BGPsec are both deployed, then there are twelve possible combinations between OV, BGPsec validation, and route-leak detection outcomes. As stated earlier, since BGPsec protects the RLP encoding, there would be added certainty in route-leak detection outcome if an update is BGPsec valid (see Section 3.2).

3.4. Are there cases when valley-free violations can be considered legitimate?

There are studies in the literature [Anwar] [Giotsas] [Wijchers] observing and analyzing the behavior of routes announced in BGP updates using data gathered from the Internet. The studies have

focused on how often there appear to be valley-free (e.g., Gao-Rexford [Gao] model) violations, and if they can be explained [Anwar]. One important consideration for explanation of the violations is per-prefix routing policies, i.e., routes for prefixes with different peering relationships may be sent over the same link. One encouraging result reported in [Anwar] is that when per-prefix routing policies are taken into consideration in the data analysis, more than 80% of the observed routing decisions fit the valley-free model (see Section 4.3 and SPA-1 data in Figure 2). [Anwar] also observes, "it is well known that this model [the basic Gao-Rexford model and some variations of it] fails to capture many aspects of the interdomain routing system. These aspects include AS relationships that vary based on the geographic region or destination prefix, and traffic engineering via hot-potato routing or load balancing." So, there may be potential for explaining the remaining (20% or less) violations of valley-free as well.

One major design factor is that the Route-Leak Protection (RLP) encoding is per prefix. Hence, the solution is consistent with ISPs' per-prefix routing policies. Large global and other major ISPs will be the likely early adopters, and they are expected to have expertise in setting policies (including per prefix policies, if applicable), and make proper use of the RLP indications on a per prefix basis. When the large ISPs participate in this solution deployment, it is envisioned that they would form a ring of protection against route leaks, and co-operatively avoid many of the common types of route leaks that are observed. Route leaks may still happen occasionally within the customer cones (if some customer ASes are not participating or not diligently implementing RLP), but such leaks are unlikely to propagate from one large participating ISP to another.

3.5. Comparison with other methods (routing security BCPs)

It is reasonable to ask if techniques considered in BCPs such as[RFC7454] (BGP Operations and Security) and [NIST-800-54] may be adequate to address route leaks. The prefix filtering recommendations in the BCPs may be complementary but not adequate. The difficulty is in ISPs' ability to construct prefix filters that represent their customer cones (CC) accurately, especially when there are many levels in the hierarchy within the CC. In the RLP-encoding based solution described here, each AS sets RLP for each route propagated and thus signals if it must not be subsequently propagated to a transit provider or peer.

AS path based Outbound Route Filter (ORF) described in [I-D.ietf-idr-aspath-orf] is also an interesting complementary technique. It can be used as an automated collaborative messaging system (implemented in BGP) for ISPs to try to develop a complete

view of the ASes and AS paths in their CCs. Once an ISP has that view, then AS path filters can be possibly used to detect route leaks. One limitation of this technique is that it cannot duly take into account the fact that routes for prefixes with different peering relationships may be sent over the same link between ASes. Also, the success of AS path based ORF depends on whether ASes at all levels of the hierarchy in a CC participate and provide accurate information (in the ORF messages) about the AS paths they expect to have in their BGP updates.

3.6. Per-Hop RLP Field or Single RLP Flag per Update?

The route-leak detection and mitigation mechanism described in [I-D.ietf-idr-route-leak-detection-mitigation] is based on setting RLP Fields on a per-hop basis. There is another possible mechanism based on a single RLP flag per update.

Method A - Per-Hop RLP Field: The sender (eBGP router) on each hop in the AS path sets its RLP Field = 1 if sending the update to a customer or lateral peer (see Section 3.2 in [I-D.ietf-idr-route-leak-detection-mitigation]). No AS (if operating correctly) would rewrite the RLP Field set by any preceding AS.

Method Z - Single RLP Flag per Update: As it propagates, the update would have at most one RLP flag. Once an eBGP router (in the update path) determines that it is sending an update towards a customer or lateral peer AS, it sets the RLP flag. The flag value equals the AS number of the eBGP router that is setting it. Once the flag is set, subsequent ASes in the path must propagate the flag as is.

To compare Methods A and Z, consider the example illustrated in Figure 1. Consider a partial deployment scenario in which AS1, AS2, AS3 and AS5 participate in RLP, and AS4 does not. AS1 (2 levels deep in AS3’s customer cone) has imperfect RLP operation. Each complying AS’s route leak mitigation policy is to prefer an update not marked as route leak (see Section 3.3 in [I-D.ietf-idr-route-leak-detection-mitigation]). If there is no alternative, then a transit-provider may accept and propagate a marked update from a customer to avoid unreachability. In this example, multi-homed AS4 leaks a route received for prefix Q from transit-provider AS3 to transit-provider AS5.

```
                        +-----------+  RLP=1    +-----------+
                        |    AS3    |---------->|    AS5    |
                        |(Major ISP)|        U2 |(Major ISP)|
                        +-----------+           +-----------+
                           /\       \            /\ U1
    Route for Q propagated /         \RLP=1      /
      due to lack of      /RLP=0      \         / (route leak;
     alternate route     /            \/       /   bad behavior)
                 +---------+       +-------------+
                 |   AS2   |       |     AS4     |
                 +---------+       +-------------+
                     /\             (legacy; does not support RLP)
                     /
                    /
                   /RLP=1 (set incorrectly)
                  /
        +----------+
        |    AS1   |
        +----------+
           /\
           /
          / Prefix Q
```

Figure 1: Example for comparison of Method A vs. Method Z

If Method A is implemented in the network, the two BGP updates for prefix Q received at AS5 are (note that AS4 is not participating in RLP):

U1A: Q [AS4 AS3 AS2 AS1] {RLP3(AS3)=1, RLP2(AS2)=0, RLP1(AS1)=1} ..... from AS4

U2A: Q [AS3 AS2 AS1] {RLP3(AS3)=1, RLP2(AS2)=0, RLP1(AS1)=1} ..... from AS3

Alternatively, if Method Z is implemented in the network, the two BGP updates for prefix Q received at AS5 are:

U1B: Q [AS4 AS3 AS2 AS1] {RLP(AS1)=1} ..... from AS4

U2B: Q [AS3 AS2 AS1] {RLP(AS1)=1} ..... from AS3

All received routes for prefix Q at AS5 are marked as route leak in either case (Method A or Z). In the case of Method A, AS5 can use additional information gleaned from the RLP fields in the updates to possibly make a better best path selection. For example, AS5 can determine that U1A update received from its customer AS4 exhibits

violation of two RLP fields (those set by AS1 and AS3) and one of them was set just two hops away. But U2A update exhibits that only one RLP field was violated and that was set three hops back. Based on this logic, AS5 may prefer U2A over U1A (even though U1A is a customer route). This would be a good decision. However, Method Z does not facilitate this kind of more rational decision process. With Method Z, both updates U1B and U2B exhibit that they violated only one RLP field (set by AS1 several hops away). AS5 may then prefer U1B over U2B since U1B is from a customer, and that would be bad decision. This illustrates that, due to more information in per-hop RLP Fields, Method A seems to be operationally more beneficial than Method Z.

Further, for detection and notification of neighbor AS's non-compliance, Method A (per-hop RLP) is better than Method Z (single RLP). With Method A, the bad behavior of AS4 would be explicitly evident to AS5 since it violated AS3's (only two hops away) RLP field as well. AS5 would alert AS4 and AS2 would alert AS1 about lack of compliance (when Method A is used). With Method Z, the alerting process may not be as expeditious.

3.7. Prevention of Route Leaks at Local AS: Intra-AS Messaging

Note: The intra-AS messaging for route leak prevention can be done using a non-transitive BGP Community or Attribute. The Community-based method is described below. For the BGP Attribute-based method, see [I-D.ietf-idr-bgp-open-policy].

3.7.1. Non-Transitive BGP Community for Intra-AS Messaging

The following procedure (or similar) for intra-AS messaging (i.e., between ingress and egress routers) for prevention of route leaks is a fairly common practice used by large ISPs. (Note: This information was gathered from discussions on the NANOG mailing list [Nanog-thread-June2016] as well as through private discussions with operators of large ISP networks.)

Routes are tagged on ingress to an AS with communities for origin, including the type of eBGP peer it was learned from (customer, provider or lateral peer), geographic location, etc. The community attributes are carried across the AS with the routes. These communities are used along with additional logic in route policies to determine which routes are to be announced to which eBGP peers and which are to be dropped. In this process, the ISP's AS also ensures that routes learned from a transit-provider or a lateral peer (i.e., non-transit) at an ingress router are not leaked at an egress router to another transit-provider or lateral peer.

Additionally, in many cases, ISP network operators' outbound policies require explicit matches for expected communities before passing routes. This helps ensure that that if an update has been entered into the RIB-in but has missed its ingress community tagging (due to a missing/misapplied ingress policy), it will not be inadvertently leaked.

The above procedure (or a simplified version of it) is also applicable when an AS consists of a single eBGP router. It is recommended that all AS operators SHOULD implement the procedure described above (or similar that is appropriate for their network) to prevent route leaks that they have direct control over.

3.8. Stopgap Solution when Only Origin Validation is Deployed

A stopgap method is described here for detection and mitigation of route leaks for the intermediate phase when OV is deployed but BGP protocol on the wire is unchanged. The stopgap solution can be in the form of construction of a prefix filter list from ROAs. A suggested procedure for constructing such a list comprises of the following steps:

- ISP makes a list of all the ASes (Cust_AS_List) that are in its customer cone (ISP's own AS is also included in the list). (Some of the ASes in Cust_AS_List may be multi-homed to another ISP and that is OK.)

- ISP downloads from the RPKI repositories a complete list (Cust_ROA_List) of valid ROAs that contain any of the ASes in Cust_AS_List.

- ISP creates a list of all the prefixes (Cust_Prfx_List) that are contained in any of the ROAs in Cust_ROA_List.

- Cust_Prfx_List is the allowed list of prefixes that is permitted by the ISP's AS, and will be forwarded by the ISP to upstream ISPs, customers, and peers.

- A route for a prefix that is not in Cust_Prfx_List but announced by one of ISP's customers is 'marked' as a potential route leak. Further, the ISP's router SHOULD prefer an alternate route that is Valid (i.e., valid according to origin validation) and 'clean' (i.e., not marked) over the 'marked' route. The alternate route may be from a peer, transit provider, or different customer.

Special considerations regarding the above procedure may be needed for DDoS mitigation service providers. They typically originate or announce a DDoS victim's prefix to their own ISP on a short notice

during a DDoS emergency. Some provisions would need to be made for such cases, and they can be determined with the help of inputs from DDoS mitigation service providers.

For developing a list of all the ASes (Cust_AS_List) that are in the customer cone of an ISP, the AS path based Outbound Route Filter (ORF) technique [I-D.ietf-idr-aspath-orf] can be helpful (see discussion in Section 3.5).

Another technique based on AS_PATH filters is described in [Snijders]. This method is applicable to very large ISPs that have lateral peering. For a pair of such very large ISPs, say A and B, the method depends on ISP A communicating out-of-band (e.g., by email) with ISP B about whether or not it (ISP A) has any transit providers. This out-of-band knowledge enables ISP B to apply suitable AS_PATH filtering criteria for routes involving the presence of ISP A in the path and prevent certain kinds of route leaks (see [Snijders] for details).

4. Security Considerations

This document requires no security considerations. See [I-D.ietf-idr-route-leak-detection-mitigation] for security considerations for the solution for route leaks detection and mitigation.

5. IANA Considerations

This document has no IANA actions.

6. References

6.1. Normative References

[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>.

6.2. Informative References

[Anwar] Anwar, R., Niaz, H., Choffnes, D., Cunha, I., Gill, P., and N. Katz-Bassett, "Investigating Interdomain Routing Policies in the Wild", ACM Internet Measurement Conference (IMC), October 2015, <http://www.cs.usc.edu/assets/007/94928.pdf>.

[draft-dickson-sidr-route-leak-solns]
Dickson, B., "Route Leaks -- Proposed Solutions", IETF Internet Draft (expired), March 2012, <https://tools.ietf.org/html/draft-dickson-sidr-route-leak-solns-01>.

[draft-kunzinger-idrp-ISO10747-01]
Kunzinger, C., "Inter-Domain Routing Protocol (IDRP)", IETF Internet Draft (expired), November 1994, <https://tools.ietf.org/pdf/draft-kunzinger-idrp-ISO10747-01.pdf>.

[Gao] Gao, L. and J. Rexford, "Stable Internet routing without global coordination", IEEE/ACM Transactions on Networking, December 2001, <http://www.cs.princeton.edu/~jrex/papers/sigmetrics00.long.pdf>.

[Gao-Rexford]
Freedman, M., "Interdomain Routing Policy", Princeton University COS 461 Lecture Notes; Slides 25-27, Spring 2011, <http://www.cs.princeton.edu/courses/archive/spr11/cos461/docs/lec17-bgp-policy.ppt>.

[Gill] Gill, P., Schapira, M., and S. Goldberg, "A Survey of Interdomain Routing Policies", ACM SIGCOMM Computer Communication Review, January 2014, <https://www.cs.bu.edu/~goldbe/papers/survey.pdf>.

[Giotsas] Giotsas, V. and S. Zhou, "Valley-free violation in Internet routing - Analysis based on BGP Community data", IEEE ICC 2012, June 2012.

[I-D.ietf-idr-aspath-orf]
Hares, S. and K. Patel, "AS Path Based Outbound Route Filter for BGP-4", draft-ietf-idr-aspath-orf-13 (work in progress), December 2016.

[I-D.ietf-idr-bgp-open-policy]
Azimov, A., Bogomazov, E., Bush, R., Patel, K., and K. Sriram, "Route Leak Prevention using Roles in Update and Open messages", draft-ietf-idr-bgp-open-policy-03 (work in progress), June 2018.

[I-D.ietf-idr-route-leak-detection-mitigation]
Sriram, K., Montgomery, D., Dickson, B., Patel, K., and A. Robachevsky, "Methods for Detection and Mitigation of BGP Route Leaks", draft-ietf-idr-route-leak-detection-mitigation-08 (work in progress), March 2018.

[Luckie] Luckie, M., Huffaker, B., Dhamdhere, A., Giotsas, V., and kc. claffy, "AS Relationships, Customer Cones, and Validation", IMC 2013, October 2013, <http://www.caida.org/~amogh/papers/asrank-IMC13.pdf>.

[Nanog-thread-June2016]
"Intra-AS messaging for route leak prevention", NANOG Email List - Discussion Thread , June 2016, <http://mailman.nanog.org/pipermail/nanog/2016-June/thread.html#86348>.

[NIST-800-54]
Kuhn, D., Sriram, K., and D. Montgomery, "Border Gateway Protocol Security", NIST Special Publication 800-54, July 2007, <http://csrc.nist.gov/publications/nistpubs/800-54/SP800-54.pdf>.

[proceedings-sixth-ietf]
Gross, P., "Proceedings of the April 22-24, 1987 Internet Engineering Task Force", April 1987, <https://www.ietf.org/proceedings/06.pdf>.

[RFC1105-obsolete]
Lougheed, K. and Y. Rekhter, "A Border Gateway Protocol (BGP)", IETF RFC (obsolete), June 1989, <https://tools.ietf.org/html/rfc1105>.

[RFC7454] Durand, J., Pepelnjak, I., and G. Doering, "BGP Operations and Security", BCP 194, RFC 7454, DOI 10.17487/RFC7454, February 2015, <https://www.rfc-editor.org/info/rfc7454>.

[RFC7908] Sriram, K., Montgomery, D., McPherson, D., Osterweil, E., and B. Dickson, "Problem Definition and Classification of BGP Route Leaks", RFC 7908, DOI 10.17487/RFC7908, June 2016, <https://www.rfc-editor.org/info/rfc7908>.

[RFC8205] Lepinski, M., Ed. and K. Sriram, Ed., "BGPsec Protocol Specification", RFC 8205, DOI 10.17487/RFC8205, September 2017, <https://www.rfc-editor.org/info/rfc8205>.

[Snijders]
Snijders, J., "Practical everyday BGP filtering with AS_PATH filters: Peer Locking", NANOG-47 Chicago, IL, USA, June 2016, <https://www.nanog.org/sites/default/files/Snijders_Everyday_Practical_Bgp.pdf>.

[Wijchers]
Wijchers, B. and B. Overeinder, "Quantitative Analysis of BGP Route Leaks", RIPE-69, November 2014, <https://ripe69.ripe.net/presentations/157-RIPE-69-Routing-WG.pdf>.

Acknowledgements

The authors wish to thank Jared Mauch, Jeff Haas, Job Snijders, Warren Kumari, Amogh Dhamdhere, Jakob Heitz, Geoff Huston, Randy Bush, Alexander Azimov, Ruediger Volk, Sue Hares, Wes George, Job Snijders, Chris Morrow, Sandy Murphy, Danny McPherson, and Eric Osterweil for comments, suggestions, and critique. The authors are also thankful to Padma Krishnaswamy, Oliver Borchert, and Okhee Kim for their review and comments.

Contributors

The following people made significant contributions to this document and should be considered co-authors:

Alexander Azimov
Qrator Labs
Email: aa@qrator.net

Brian Dickson
Independent
Email: brian.peter.dickson@gmail.com

Doug Montgomery
USA National Institute of Standards and Technology
Email: dougm@nist.gov

Keyur Patel
Arrcus
Email: keyur@arrcus.com

Andrei Robachevsky
Internet Society
Email: robachevsky@isoc.org

Eugene Bogomazov
Qrator Labs
Email: eb@qrator.net

Randy Bush
Internet Initiative Japan
Email: randy@psg.com

Author’s Address

Kotikalapudi Sriram (editor)
USA National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, MD 20899
United States of America

Email: ksriram@nist.gov

IDR Working Group
Internet-Draft
Intended status: Informational
Expires: 9 September 2023

K. Sriram, Ed.
USA NIST

8 March 2023

Design Discussion of Route Leaks Solution Methods
draft-sriram-idr-route-leak-solution-discussion-09

Abstract

This document captures the design rationale of the route leaks solution document (see draft-ietf-idr-route-leak-detection-mitigation, draft-ietf-grow-route-leak-detection-mitigation). The designers needed to balance many competing factors, and this document provides insights into the design questions and their resolution.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document captures the design rationale of the route leaks solution document [I-D.ietf-idr-route-leak-detection-mitigation] [I-D.ietf-grow-route-leak-detection-mitigation]. The designers needed to balance many competing factors, and this document provides insights into the design questions and their resolution.

2. Related Prior Work

The solution described in [I-D.ietf-idr-route-leak-detection-mitigation] is based on setting an attribute in BGP route announcement to manage the transmission/ receipt of the announcement based on the type of neighbor (e.g., customer, transit provider, etc.). Documented prior work related to this basic idea and mechanism dates back to at least the 1980's. Some examples of prior work are: (1) Information flow rules described in [proceedings-sixth-ietf] (see pp. 195-196); (2) Link Type described in [RFC1105-obsolete] (see pp. 4-5); (3) Hierarchical

Recording described in [draft-kunzinger-idrp-ISO10747-01] (see Section 6.3.1.12). The problem of route leaks [RFC7908] and possible solution mechanisms based on encoding peering-link type information, e.g., P2C (i.e., Transit-Provider to Customer), C2P (i.e., Customer to Transit-Provider), p2p (i.e., peer to peer) etc., in BGPsec [RFC8205] updates and protecting the same under BGPsec path signatures have been discussed in IETF SIDR WG at least since 2011. [draft-dickson-sidr-route-leak-solns] attempted to describe these mechanisms in a BGPsec context. The draft expired in 2012. [draft-dickson-sidr-route-leak-solns] defined neighbor relationships on a per link basis, but in [I-D.ietf-idr-route-leak-detection-mitigation] the relationship is encoded per prefix, as routes for prefixes with different peering relationships may be sent over the same link. Also [draft-dickson-sidr-route-leak-solns] proposed a second signature block for the link type encoding, separate from the path signature block in BGPsec. By contrast, in [I-D.ietf-idr-route-leak-detection-mitigation], when BGPsec-based solution is considered, cryptographic protection is provided for Route-Leak Protection (RLP) encoding using the same signature block as that for path signatures (see Section 3.2.2 in [I-D.ietf-idr-route-leak-detection-mitigation]).

3. Design Rationale and Discussion

This section provides design justifications for the solution specified in [I-D.ietf-idr-route-leak-detection-mitigation], and also answers some questions that are anticipated or have been raised in the IETF IDR and SIDR working group meetings.

3.1. Explanation of Rules 1 and 2 in the solution document

In Section 3.3 in [I-D.ietf-idr-route-leak-detection-mitigation], Rules 1 and 2 were stated and the route leak mitigation policy was based on these rules to preserve the property of stable route convergence (i.e., avoid possibility of persistent route oscillations). Rule 1 is stated as follows:

* Rule 1: If ISP A receives a route r1 from customer AS C and another route r2 from provider (or peer) AS B (for the same prefix), and both routes r1 and r2 contain AS C and AS X (any X not equal to C) in the path and contain [X] in their RLP Attributes, then prioritize the customer (AS C) route over the provider (or peer) route.

The rationale for Rule 1 can be developed as follows.

Preference condition for route stability: Prefer customer routes over peer or provider routes (see pp. 25-27 in [Gao-Rexford]).

Topology condition for route stability: No cycle of customer-provider relationships (see pp. 25-27 in [Gao-Rexford]).

Route-Leak Detection Theorem: Let it be given that ISP A receives a route r1 from customer AS C and another route r2 from provider AS B (for the same prefix), and each of the routes r1 and r2 contains AS C and AS X in its AS path and contains [X] in its RLP Attribute. Then, clearly r1 is in violation of [X]. It follows that r2 is also necessarily in violation of [X].

Proof: Let us suppose that r2 is not in violation of [X]. That implies that r2's path from C to B to A included only P2C links. That would mean that there is a cycle of customer-provider relationships involving the ASes in the AS path in r2. However, any such cycle is ruled out in practice by the topology condition for route stability as stated above. QED.

Corollary 1: The route leak detection theorem holds also when "provider AS B" in the theorem is replaced by "peer AS B". (Here peer means a lateral peer.)

Proof: Since r2 contains [X] in the RLP Attribute set by an AS prior to peer AS B, it follows that r2 is in violation of [X]. QED.

It can be observed that Rule 1 follows from the combination of the Theorem, Corollary 1 and the preference condition for route stability (stated above).

In Section 3.3 in [I-D.ietf-idr-route-leak-detection-mitigation], Rule 2 is stated as follows:

* Rule 2: If ISP A receives a route r1 from peer AS C and another route r2 from provider AS B (for the same prefix), and both routes r1 and r2 contain AS C and AS X (any X not equal to C) in the path and contain [X] in their RLP Attributes, then prioritize the peer (AS C) route over the provider (AS B) route.

The rationale for Rule 2 can be developed as follows.

Corollary 2: The route leak detection theorem holds also when "customer AS C" in the theorem is replaced by "peer AS C".

Proof for Corollary 2: Let us suppose that r2 is not in violation of [X]. That implies that r2's path from C to B to A included only P2C links. This results in a topology in which A's lateral peer B is

also A's transit provider's transit provider. This gives rise to possibility of looping of routes since A can send routes to its transit B, B can forward the routes to its transit C, and C can forward the routes to its peer A. But such looping is forbidden by the topology condition stated above.

It can be observed that Rule 2 follows from Corollary 2. In essence, if the provider route (r2) is a detoured (longer) version of the lateral peer route (r1), and violates the same RLP [X] as does the peer route, then prefer the shorter route (r2) via the peer.

Rules 1 and 2 are

3.2. Is route-leak solution without cryptographic protection an attack vector?

It has been asked if a route-leak solution without BGPsec, i.e., when RLP Fields are not protected, can turn into a new attack vector. The answer seems to be: not really! Even the NLRI and AS_PATH in BGP updates are attack vectors, and RPKI/OV/BGPsec seek to fix that. Consider the following. Say, if 99% of route leaks are accidental and 1% are malicious, and if route-leak solution without BGPsec eliminates the 99%, then perhaps it is worth it (step in the right direction). When BGPsec comes into deployment, the route-leak protection (RLP) bits can be mapped into BGPsec (using the Flags field) and then necessary security will be in place as well (within each BGPsec island as and when they emerge).

Further, let us consider the worst-case damage that can be caused by maliciously manipulating the RLP Field values in an implementation without cryptographic protection (i.e., sans BGPsec). Manipulation of the RLP bits can result in one of two types of attacks: (a) Upgrade attack and (b) Downgrade attack. Descriptions and discussions about these attacks follow. In what follows, P2C stands for transit provider to customer (Down); C2P stands for customer to transit provider (Up), and p2p stands for peer to peer (lateral or non-transit relationship).

(a) Upgrade attack: An AS that wants to intentionally leak a route would alter the RLP encodings for the preceding hops from 1 (i.e., 'Do not Propagate Up or Lateral') to 0 (default) wherever applicable. This poses no problem for a route that keeps propagating in the 'Down' (P2C) direction. However, for a route that propagates 'Up' (C2P) or 'Lateral' (p2p), the worst that can happen is that a route leak goes undetected. That is, a receiving router would not be able to detect the leak for the route in question by the RLP mechanism described here. However, the receiving router may still detect and mitigate it in some cases by applying other means such as prefix

filters [RFC7454] [NIST-800-54]. If some malicious leaks go undetected (when RLP is deployed without BGPsec) that is possibly a small price to pay for the ability to detect the bulk of route leaks that are accidental.

(b) Downgrade attack: RLP encoding is set to 1 (i.e., 'Do not Propagate Up or Lateral') when it should be set to 0 (default). This would result in a route being mis-detected and marked as a route leak. By default, RLP encoding is set to 0, and that helps reduce errors of this kind (i.e., accidental downgrade incidents). Every AS or ISP wants reachability for prefixes it originates and for its customer prefixes. So, an AS or ISP is not likely to change an RLP value 0 to 1 intentionally. If a route leak is detected (due to intentional or accidental downgrade) by a receiving router, it would prefer an alternate 'clean' route from a transit provider or peer over a 'marked' route from a customer. It may end up with a suboptimal path. In order to have reachability, the receiving router would accept a 'marked' route if there is no alternative that is 'clean'. So, RLP downgrade attacks (intentional or accidental) would be quite rare, and the consequences do not appear to be grave.

3.3. Combining results of route-leak detection, OV and BGPsec validation for path selection decision

Combining the results of route-leak detection, OV, and BGPsec validation for path selection decision is up to local policy in a receiving router. As an example, a router may always give precedence to outcomes of OV and BGPsec validation over that of route-leak detection. That is, if an update fails OV or BGPsec validation, then the update is not considered a candidate for path selection. Instead, an alternate update is chosen that passed OV and BGPsec validation and additionally was not marked as route leak.

If only OV is deployed (and not BGPsec), then there are six possible combinations between OV and route-leak detection outcomes. Because there are three possible outcomes for OV (NotFound, Valid, and Invalid) and two possible outcomes for route-leak detection (marked as leak and not marked). If OV and BGPsec are both deployed, then there are twelve possible combinations between OV, BGPsec validation, and route-leak detection outcomes. As stated earlier, since BGPsec protects the RLP encoding, there would be added certainty in route-leak detection outcome if an update is BGPsec valid (see Section 3.2).

3.4. Are there cases when valley-free violations can be considered legitimate?

There are studies in the literature [Anwar] [Giotsas] [Wijchers] observing and analyzing the behavior of routes announced in BGP updates using data gathered from the Internet. The studies have focused on how often there appear to be valley-free (e.g., Gao-Rexford [Gao] model) violations, and if they can be explained [Anwar]. One important consideration for explanation of the violations is per-prefix routing policies, i.e., routes for prefixes with different peering relationships may be sent over the same link. One encouraging result reported in [Anwar] is that when per-prefix routing policies are taken into consideration in the data analysis, more than 80% of the observed routing decisions fit the valley-free model (see Section 4.3 and SPA-1 data in Figure 2). [Anwar] also observes, "it is well known that this model [the basic Gao-Rexford model and some variations of it] fails to capture many aspects of the interdomain routing system. These aspects include AS relationships that vary based on the geographic region or destination prefix, and traffic engineering via hot-potato routing or load balancing." So, there may be potential for explaining the remaining (20% or less) violations of valley-free as well.

One major design factor is that the Route-Leak Protection (RLP) encoding is per prefix. Hence, the solution is consistent with ISPs' per-prefix routing policies. Large global and other major ISPs will be the likely early adopters, and they are expected to have expertise in setting policies (including per prefix policies, if applicable), and make proper use of the RLP indications on a per prefix basis. When the large ISPs participate in this solution deployment, it is envisioned that they would form a ring of protection against route leaks, and co-operatively avoid many of the common types of route leaks that are observed. Route leaks may still happen occasionally within the customer cones (if some customer ASes are not participating or not diligently implementing RLP), but such leaks are unlikely to propagate from one large participating ISP to another.

3.5. Comparison with other methods (routing security BCPs)

It is reasonable to ask if techniques considered in BCPs such as[RFC7454] (BGP Operations and Security) and [NIST-800-54] may be adequate to address route leaks. The prefix filtering recommendations in the BCPs may be complementary but not adequate. The difficulty is in ISPs' ability to construct prefix filters that represent their customer cones (CC) accurately, especially when there are many levels in the hierarchy within the CC. In the RLP-encoding based solution described here, each AS sets RLP for each route propagated and thus signals if it must not be subsequently propagated

to a transit provider or peer.

AS path based Outbound Route Filter (ORF) described in [I-D.ietf-idr-aspath-orf] is also an interesting complementary technique. It can be used as an automated collaborative messaging system (implemented in BGP) for ISPs to try to develop a complete view of the ASes and AS paths in their CCs. Once an ISP has that view, then AS path filters can be possibly used to detect route leaks. One limitation of this technique is that it cannot duly take into account the fact that routes for prefixes with different peering relationships may be sent over the same link between ASes. Also, the success of AS path based ORF depends on whether ASes at all levels of the hierarchy in a CC participate and provide accurate information (in the ORF messages) about the AS paths they expect to have in their BGP updates.

3.6. Per-Hop RLP Field or Single RLP Flag per Update?

The route-leak detection and mitigation mechanism described in [I-D.ietf-idr-route-leak-detection-mitigation] is based on setting RLP Fields on a per-hop basis. There is another possible mechanism based on a single RLP flag per update.

Method A - Per-Hop RLP Field: The sender (eBGP router) on each hop in the AS path sets its RLP Field = 1 if sending the update to a customer or lateral peer (see Section 3.2 in [I-D.ietf-idr-route-leak-detection-mitigation]). No AS (if operating correctly) would rewrite the RLP Field set by any preceding AS.

Method Z - Single RLP Flag per Update: As it propagates, the update would have at most one RLP flag. Once an eBGP router (in the update path) determines that it is sending an update towards a customer or lateral peer AS, it sets the RLP flag. The flag value equals the AS number of the eBGP router that is setting it. Once the flag is set, subsequent ASes in the path must propagate the flag as is.

To compare Methods A and Z, consider the example illustrated in Figure 1. Consider a partial deployment scenario in which AS1, AS2, AS3 and AS5 participate in RLP, and AS4 does not. AS1 (2 levels deep in AS3's customer cone) has imperfect RLP operation. Each complying AS's route leak mitigation policy is to prefer an update not marked as route leak (see Section 3.3 in [I-D.ietf-idr-route-leak-detection-mitigation]). If there is no alternative, then a transit-provider may accept and propagate a marked update from a customer to avoid unreachability. In this example, multi-homed AS4 leaks a route received for prefix Q from transit-provider AS3 to transit-provider AS5.

```
                        +-----------+  RLP=1    +-----------+
                        |    AS3    |---------->|    AS5    |
                        |(Major ISP)|        U2 |(Major ISP)|
                        +-----------+           +-----------+
                           /\       \            /\ U1
    Route for Q propagated /         \RLP=1      /
      due to lack of      /RLP=0      \         / (route leak;
     alternate route     /            \/       /   bad behavior)
                +---------+       +-------------+
                |   AS2   |       |     AS4     |
                +---------+       +-------------+
                    /\             (legacy; does not support RLP)
                    /
                   /
                  /RLP=1 (set incorrectly)
                 /
        +----------+
        |    AS1   |
        +----------+
           /\
           /
          / Prefix Q
```

Figure 1: Example for comparison of Method A vs. Method Z

If Method A is implemented in the network, the two BGP updates for prefix Q received at AS5 are (note that AS4 is not participating in RLP):

U1A: Q [AS4 AS3 AS2 AS1] {RLP3(AS3)=1, RLP2(AS2)=0, RLP1(AS1)=1} ..... from AS4

U2A: Q [AS3 AS2 AS1] {RLP3(AS3)=1, RLP2(AS2)=0, RLP1(AS1)=1} ..... from AS3

Alternatively, if Method Z is implemented in the network, the two BGP updates for prefix Q received at AS5 are:

U1B: Q [AS4 AS3 AS2 AS1] {RLP(AS1)=1} ..... from AS4

U2B: Q [AS3 AS2 AS1] {RLP(AS1)=1} ..... from AS3

All received routes for prefix Q at AS5 are marked as route leak in either case (Method A or Z). In the case of Method A, AS5 can use additional information gleaned from the RLP fields in the updates to possibly make a better best path selection. For example, AS5 can determine that U1A update received from its customer AS4 exhibits violation of two RLP fields (those set by AS1 and AS3) and one of

them was set just two hops away. But U2A update exhibits that only one RLP field was violated and that was set three hops back. Based on this logic, AS5 may prefer U2A over U1A (even though U1A is a customer route). This would be a good decision. However, Method Z does not facilitate this kind of more rational decision process. With Method Z, both updates U1B and U2B exhibit that they violated only one RLP field (set by AS1 several hops away). AS5 may then prefer U1B over U2B since U1B is from a customer, and that would be bad decision. This illustrates that, due to more information in per-hop RLP Fields, Method A seems to be operationally more beneficial than Method Z.

Further, for detection and notification of neighbor AS's non-compliance, Method A (per-hop RLP) is better than Method Z (single RLP). With Method A, the bad behavior of AS4 would be explicitly evident to AS5 since it violated AS3's (only two hops away) RLP field as well. AS5 would alert AS4 and AS2 would alert AS1 about lack of compliance (when Method A is used). With Method Z, the alerting process may not be as expeditious.

3.7. Prevention of Route Leaks at Local AS: Intra-AS Messaging

Note: The intra-AS messaging for route leak prevention can be done using a non-transitive BGP Community or Attribute. The Community-based method is described below. For the BGP Attribute-based method, see [RFC9234].

3.7.1. Non-Transitive BGP Community for Intra-AS Messaging

The following procedure (or similar) for intra-AS messaging (i.e., between ingress and egress routers) for prevention of route leaks is a fairly common practice used by large ISPs. (Note: This information was gathered from discussions on the NANOG mailing list [Nanog-thread-June2016] as well as through private discussions with operators of large ISP networks.)

Routes are tagged on ingress to an AS with communities for origin, including the type of eBGP peer it was learned from (customer, provider or lateral peer), geographic location, etc. The community attributes are carried across the AS with the routes. These communities are used along with additional logic in route policies to determine which routes are to be announced to which eBGP peers and which are to be dropped. In this process, the ISP's AS also ensures that routes learned from a transit-provider or a lateral peer (i.e., non-transit) at an ingress router are not leaked at an egress router to another transit-provider or lateral peer.

Additionally, in many cases, ISP network operators' outbound policies require explicit matches for expected communities before passing routes. This helps ensure that that if an update has been entered into the RIB-in but has missed its ingress community tagging (due to a missing/misapplied ingress policy), it will not be inadvertently leaked.

The above procedure (or a simplified version of it) is also applicable when an AS consists of a single eBGP router. It is recommended that all AS operators SHOULD implement the procedure described above (or similar that is appropriate for their network) to prevent route leaks that they have direct control over.

3.8. Stopgap Solution when Only Origin Validation is Deployed

A stopgap method is described here for detection and mitigation of route leaks for the intermediate phase when OV is deployed but BGP protocol on the wire is unchanged. The stopgap solution can be in the form of construction of a prefix filter list from ROAs. A suggested procedure for constructing such a list comprises of the following steps:

* ISP makes a list of all the ASes (Cust_AS_List) that are in its customer cone (ISP's own AS is also included in the list). (Some of the ASes in Cust_AS_List may be multi-homed to another ISP and that is OK.)

* ISP downloads from the RPKI repositories a complete list (Cust_ROA_List) of valid ROAs that contain any of the ASes in Cust_AS_List.

* ISP creates a list of all the prefixes (Cust_Prfx_List) that are contained in any of the ROAs in Cust_ROA_List.

* Cust_Prfx_List is the allowed list of prefixes that is permitted by the ISP's AS, and will be forwarded by the ISP to upstream ISPs, customers, and peers.

* A route for a prefix that is not in Cust_Prfx_List but announced by one of ISP's customers is 'marked' as a potential route leak. Further, the ISP's router SHOULD prefer an alternate route that is Valid (i.e., valid according to origin validation) and 'clean' (i.e., not marked) over the 'marked' route. The alternate route may be from a peer, transit provider, or different customer.

Special considerations regarding the above procedure may be needed for DDoS mitigation service providers. They typically originate or announce a DDoS victim's prefix to their own ISP on a short notice

during a DDoS emergency. Some provisions would need to be made for such cases, and they can be determined with the help of inputs from DDoS mitigation service providers.

For developing a list of all the ASes (Cust_AS_List) that are in the customer cone of an ISP, the AS path based Outbound Route Filter (ORF) technique [I-D.ietf-idr-aspath-orf] can be helpful (see discussion in Section 3.5).

Another technique based on AS_PATH filters is described in [Snijders]. This method is applicable to very large ISPs that have lateral peering. For a pair of such very large ISPs, say A and B, the method depends on ISP A communicating out-of-band (e.g., by email) with ISP B about whether or not it (ISP A) has any transit providers. This out-of-band knowledge enables ISP B to apply suitable AS_PATH filtering criteria for routes involving the presence of ISP A in the path and prevent certain kinds of route leaks (see [Snijders] for details).

4. Security Considerations

This document requires no security considerations. See [I-D.ietf-idr-route-leak-detection-mitigation] for security considerations for the solution for route leaks detection and mitigation.

5. IANA Considerations

This document has no IANA actions.

6. References

6.1. Normative References

[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>.

6.2. Informative References

[Anwar] Anwar, R., Niaz, H., Choffnes, D., Cunha, I., Gill, P., and N. Katz-Bassett, "Investigating Interdomain Routing Policies in the Wild", ACM Internet Measurement Conference (IMC), October 2015, <http://www.cs.usc.edu/assets/007/94928.pdf>.

[draft-dickson-sidr-route-leak-solns]
Dickson, B., "Route Leaks -- Proposed Solutions", IETF Internet Draft (expired), March 2012, <https://tools.ietf.org/html/draft-dickson-sidr-route-leak-solns-01>.

[draft-kunzinger-idrp-ISO10747-01]
Kunzinger, C., "Inter-Domain Routing Protocol (IDRP)", IETF Internet Draft (expired), November 1994, <https://tools.ietf.org/pdf/draft-kunzinger-idrp-ISO10747-01.pdf>.

[Gao] Gao, L. and J. Rexford, "Stable Internet routing without global coordination", IEEE/ACM Transactions on Networking, December 2001, <http://www.cs.princeton.edu/~jrex/papers/sigmetrics00.long.pdf>.

[Gao-Rexford]
Freedman, M., "Interdomain Routing Policy", Princeton University COS 461 Lecture Notes; Slides 25-27, <http://www.cs.princeton.edu/courses/archive/spr11/cos461/docs/lec17-bgp-policy.ppt>.

[Gill] Gill, P., Schapira, M., and S. Goldberg, "A Survey of Interdomain Routing Policies", ACM SIGCOMM Computer Communication Review, January 2014, <https://www.cs.bu.edu/~goldbe/papers/survey.pdf>.

[Giotsas] Giotsas, V. and S. Zhou, "Valley-free violation in Internet routing - Analysis based on BGP Community data", IEEE ICC 2012, June 2012.

[I-D.ietf-grow-route-leak-detection-mitigation]
Sriram, K. and A. Azimov, "Methods for Detection and Mitigation of BGP Route Leaks", Work in Progress, Internet-Draft, draft-ietf-grow-route-leak-detection-mitigation-08, 24 October 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-grow-route-leak-detection-mitigation-08>.

[I-D.ietf-idr-aspath-orf]
Hares, S. and K. Patel, "AS Path Based Outbound Route Filter for BGP-4", Work in Progress, Internet-Draft, draft-ietf-idr-aspath-orf-13, 19 December 2016, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-aspath-orf-13>.

[I-D.ietf-idr-route-leak-detection-mitigation]
Sriram, K. and A. Azimov, "Methods for Detection and Mitigation of BGP Route Leaks", Work in Progress, Internet-Draft, draft-ietf-idr-route-leak-detection-mitigation-11, 18 April 2019, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-route-leak-detection-mitigation-11>.

[Luckie] Luckie, M., Huffaker, B., Dhamdhere, A., Giotsas, V., and kc. claffy, "AS Relationships, Customer Cones, and Validation", IMC 2013, October 2013, <http://www.caida.org/~amogh/papers/asrank-IMC13.pdf>.

[Nanog-thread-June2016]
"Intra-AS messaging for route leak prevention", NANOG Email List - Discussion Thread , June 2016, <http://mailman.nanog.org/pipermail/nanog/2016-June/thread.html#86348>.

[NIST-800-54]
Kuhn, D.R., Sriram, K., and D. Montgomery, "Border Gateway Protocol Security", NIST Special Publication 800-54, July 2007, <http://csrc.nist.gov/publications/nistpubs/800-54/SP800-54.pdf>.

[proceedings-sixth-ietf]
Gross, P., "Proceedings of the April 22-24, 1987 Internet Engineering Task Force", April 1987, <https://www.ietf.org/proceedings/06.pdf>.

[RFC1105-obsolete]
Lougheed, K. and Y. Rekhter, "A Border Gateway Protocol (BGP)", IETF RFC (obsolete), June 1989, <https://tools.ietf.org/html/rfc1105>.

[RFC7454] Durand, J., Pepelnjak, I., and G. Doering, "BGP Operations and Security", BCP 194, RFC 7454, DOI 10.17487/RFC7454, February 2015, <https://www.rfc-editor.org/info/rfc7454>.

[RFC7908] Sriram, K., Montgomery, D., McPherson, D., Osterweil, E., and B. Dickson, "Problem Definition and Classification of BGP Route Leaks", RFC 7908, DOI 10.17487/RFC7908, June 2016, <https://www.rfc-editor.org/info/rfc7908>.

[RFC8205] Lepinski, M., Ed. and K. Sriram, Ed., "BGPsec Protocol Specification", RFC 8205, DOI 10.17487/RFC8205, September 2017, <https://www.rfc-editor.org/info/rfc8205>.

[RFC9234] Azimov, A., Bogomazov, E., Bush, R., Patel, K., and K. Sriram, "Route Leak Prevention and Detection Using Roles in UPDATE and OPEN Messages", RFC 9234, DOI 10.17487/RFC9234, May 2022, <https://www.rfc-editor.org/info/rfc9234>.

[Snijders] Snijders, J., "Practical everyday BGP filtering with AS_PATH filters: Peer Locking", NANOG 67 Chicago, IL, USA, June 2016, <https://www.nanog.org/sites/default/files/Snijders_Everyday_Practical_Bgp.pdf>.

[Wijchers] Wijchers, B. and B. Overeinder, "Quantitative Analysis of BGP Route Leaks", RIPE-69, November 2014, <https://ripe69.ripe.net/presentations/157-RIPE-69-Routing-WG.pdf>.

Acknowledgements

The authors wish to thank Jared Mauch, Jeff Haas, Job Snijders, Warren Kumari, Amogh Dhamdhere, Jakob Heitz, Geoff Huston, Ruediger Volk, Sue Hares, John Scudder, Wes George, Chris Morrow, Sandy Murphy, Danny McPherson, and Eric Osterweil for comments, suggestions, and critique. The authors are also thankful to Padma Krishnaswamy, Oliver Borchert, and Okhee Kim for their review and comments.

Contributors

The following people made significant contributions to this document and should be considered co-authors:

Alexander Azimov
Yandex
Email: a.e.azimov@gmail.com

Brian Dickson
Independent
Email: brian.peter.dickson@gmail.com

Doug Montgomery
USA National Institute of Standards and Technology
Email: dougm@nist.gov

Keyur Patel
Arrcus
Email: keyur@arrcus.com

Andrei Robachevsky
Internet Society
Email: robachevsky@isoc.org

Eugene Bogomazov
Qrator Labs
Email: eb@qrator.net

Randy Bush
Internet Initiative Japan
Email: randy@psg.com

Author's Address

Kotikalapudi Sriram (editor)
USA National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, MD 20899
United States of America
Email: ksriram@nist.gov

IDR Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 15, 2019

A. Wang
China Telecom
H. Chen
Huawei Technologies

August 14, 2018

BGP-LS Extension for Inter-AS Topology Retrieval
draft-wang-idr-bgpls-inter-as-topology-ext-02

Abstract

This document describes the process to build BGP-LS key parameters in Native IP multi-domain scenario and defines some new inter-AS TE related TLVs for BGP-LS to let SDN controller retrieve the network topology automatically under various environments.

Such process and extension can expand the usage of BGP-LS protocol to multi- domain, enable the network operator to collect the connection relationship between different AS domains and then calculate the overall network topology automatically based on the information provided by BGP-LS 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 February 15, 2019.

Table of Contents

1. Introduction

BGP-LS [RFC7752] describes the methodology that using BGP protocol to transfer the Link-State information. Such method can enable SDN controller to collect the underlay network topology automatically, but normally it can only get the information within one IGP domain. If the operator has more than one IGP domain, and these domains interconnect with each other, there is no general TLV within current BGP- LS to transfer the interconnect information.

Draft [I-D.ietf-idr-bgpls-segment-routing-epe] defines some extensions for exporting BGP peering node topology information (including its peers, interfaces and peering ASs) in a way that is exploitable in order to compute efficient BGP Peering Engineering policies and strategies. Such information can also be used to calculate the interconnection topology among different IGP domains, but it requires the border routers to run BGP-LS protocol to collect this information and report them to the PCE/SDN controller, which restricts the deployment flexibility of BGP-LS protocol.

This draft analysizes the situations that the PCE/SDN controller needs to get about the inter-connected information between different AS domains, defines new TLVs to extend the BGP-LS protocol to

transfer the key information related to the interconnect TE topology. After that, the SDN controller can then deduce the multi-domain topology automatically based on the information from BGP-LS protocol.

2. Conventions used in this document

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

3. Inter-AS Domain Scenarios.

Fig.1 illustrates the multi-domain scenarios that this draft discussed. Normally, SDN Controller can get the topology of IGP A and IGP B individually via the BGP-LS protocol, but it can’t get the topology connection information between these two IGP domains because there is generally no IGP protocol run on the connected links.

```
                          +-----------------+
                     +----+IP SDN Controller+----+
                     |    +-----------------+    |
                     |                           |
                     |BGP-LS                     |BGP-LS
                     |                           |
     +---------------+-----+               +-----+--------------+
     | +--+        +-++   ++-+           +-++   +|-+        +--+|
     | |S1+--------+S2+---+B1+------------+B2+---+T1+--------+T2||
     | +-++   N1   +-++   ++-+           +-++   ++++   N2   +-++|
     |   |           |     |               |     ||           | |
     |   |           |     |               |     ||           | |
     | +-++        +-++   ++-+           +-++   ++++        +-++|
     | |S4+--------+S3+---+B3+------------+B4+---+T3+--------+T4||
     | +--+        +--+   ++-+           +-++   ++-+        +--+|
     |                     |               |                     |
     |                     |               |                     |
     |       IGP A         |               |      IGP B          |
     +---------------------+               +---------------------+
```

Figure 1: Inter-AS Domain Scenarios

3.1. IS-IS/OSPF Inter-AS Native IP Scenario

When the IGP A or IGP B runs native IS-IS/OSPF protocol, the operator often redistributes the IPv4/IPv6 prefixes of interconnect links into IS-IS/OSPF protocol to ensure the inter-domain connectivity.

If the IGP runs IS-IS protocol, the redistributed link information will be carried in IP External Reachability Information TLV within

the Level 2 PDU type that defined in [RFC1195], every router within the IGP domain can deduce the redistributed router from the IS-IS LSDB.

If the IGP runs OSPF protocol,[RFC2328]defines the type 5 external LSA to transfer the external IPv4 routes; [I-D.ietf-ospf-ospfv3-lsa-extend] defines the "External-Prefix TLV" to transfer the external IPv6 routes; these LSAs have also the advertising router information that initiates the redistribute activity. Every router within IGP domain can also deduce the redistributed router from the OSPF LSDB.

For prefix information that associated with each router, BGP-LS [RFC7752] defines the Prefix NLRI which is illustrated below:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
|  Protocol-ID  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Identifier                          |
|                            (64 bits)                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//              Local Node Descriptors (variable)              //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                Prefix Descriptors (variable)                //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: The IPv4/IPv6 Topology Prefix NLRI Format

For these redistributed inter-domain links, their prefix information should be included in the "Prefix Descriptor", and the associated redistributed router information should be included in the "Local Node Descriptors".

When such information is reported via the BGP-LS protocol, the PCE/SDN controller can construct the underlay inter-domain topology according to procedure described in section 5

3.2. IS-IS/OSPF Inter-AS TE Scenario

[RFC5316] and [RFC5392] define the IS-IS and OSPF extensions respectively to deal with the requirements for inter-AS traffic engineering. They define some new sub-TLVs(Remote AS Number、IPv4 Remote ASBR ID、IPv6 Remote ASBR ID) which are associated with the inter-AS TE link TLVs to report the TE topology between different domains.

These TLVs are flooded within the IGP domain automatically. If the PCE/SDN controller can know these information via one of the interior router that runs BGP-LS protocol, the PCE/SDN controller can rebuild the inter-AS TE topology correctly.

4. Inter-AS TE related TLVs

This draft proposes to add three new TLVs that is included within the inter-AS TE link NLRI to transfer the information via BGP-LS, which are required to build the inter-AS related topology by the PCE/SDN controller.

The following Link Descriptor TLVs are added into the Link NLRI in BGP-LS protocol :

TLV Code Point	Description	IS-IS/OSPF TLV /Sub-TLV	Reference (RFC/Section)
TBD	Remote AS Number	24/21	[RFC5316]/3.3.1
			[RFC5392]/3.3.1
TBD	IPv4 Remote ASBR ID	25/22	[RFC5316]/3.3.2
			[RFC5392]/3.3.2
TBD	IPv6 Remote ASBR ID	26/24	[RFC5316]/3.3.3
			[RFC5392]/3.3.3

Detail encoding of these TLVs are synchronized with the corresponding parts in [RFC5316] and [RFC5392], which keeps the BGP-LS protocol is agnostic to the underly protocol.

4.1. Remote AS Number TLV

A new TLV, the remote AS number TLV, is defined for inclusion in the link descriptor when advertising inter-AS links. The remote AS number TLV specifies the AS number of the neighboring AS to which the advertised link connects.

The remote AS number TLV is TLV type TBD (see Section 7) and is 4 octets in length. The format is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Remote AS Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Remote AS number field has 4 octets. When only 2 octets are used for the AS number, as in current deployments, the left (high-order) 2 octets MUST be set to 0. The remote AS number TLV MUST be included when a router advertises an inter-AS TE link.

4.2. IPv4 Remote ASBR ID

A new TLV, which is referred to as the IPv4 remote ASBR ID TLV, is defined for inclusion in the link descriptor when advertising inter-AS links. The IPv4 remote ASBR ID TLV specifies the IPv4 identifier of the remote ASBR to which the advertised inter-AS link connects. This could be any stable and routable IPv4 address of the remote ASBR. Use of the TE Router ID as specified in the Traffic Engineering router ID TLV [RFC5305] is RECOMMENDED.

The IPv4 remote ASBR ID TLV is TLV type TBD (see Section 7) and is 4 octets in length. The format of the IPv4 remote ASBR ID sub-TLV is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Remote ASBR ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The IPv4 remote ASBR ID TLV MUST be included if the neighboring ASBR has an IPv4 address. If the neighboring ASBR does not have an IPv4 address (not even an IPv4 TE Router ID), the IPv6 remote ASBR ID TLV MUST be included instead. An IPv4 remote ASBR ID TLV and IPv6 remote ASBR ID TLV MAY both be present in an extended IS reachability TLV.

4.3. IPv6 Remote ASBR ID

A new TLV, which is referred to as the IPv6 remote ASBR ID TLV, is defined for inclusion in the inter-AS reachability TLV when advertising inter-AS links. The IPv6 remote ASBR ID TLV specifies the IPv6 identifier of the remote ASBR to which the advertised inter-AS link connects. This could be any stable and routable IPv6 address of the remote ASBR. Use of the TE Router ID as specified in the IPv6 Traffic Engineering router ID TLV [RFC6119] is RECOMMENDED.

The IPv6 remote ASBR ID TLV is TLV type TBD (see Section 7) and is 16 octets in length. The format of the IPv6 remote ASBR ID TLV is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Remote ASBR ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Remote ASBR ID (continued)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Remote ASBR ID (continued)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Remote ASBR ID (continued)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The IPv6 remote ASBR ID TLV MUST be included if the neighboring ASBR has an IPv6 address. If the neighboring ASBR does not have an IPv6 address, the IPv4 remote ASBR ID TLV MUST be included instead. An IPv4 remote ASBR ID TLV and IPv6 remote ASBR ID TLV MAY both be present in an extended IS reachability TLV.

5. Topology Reconstruction.

When SDN Controller gets such information from BGP-LS protocol, it should compares the proximity of the redistributed prefixes. If they are under the same network scope, then it should find the corresponding associated router information, build the link between these two border routers.

After iterating the above procedures for all of the redistributed prefixes, the SDN controller can then retrieve the connection topology between different domains automatically.

6. Security Considerations

It is common for one operator to occupy several IGP domains that composited by its backbone network and several MAN(Metrio-Area-Network)s/IDCs. When they do traffic engineering from end to end that spans MAN-backbone-IDC, they need to know the inter-as topology via the process described in this draft. Then it is naturally to redistribute the interconnection prefixes in Native IP scenario.

If these IGP domains belong to different operators, it is uncommon do inter-as traffic engineering under one PCE/SDN controller, then it is unnecessary to get the inter-as topology. But redistributing the interconnection prefixes will do no harm to their networks, because the redistributed interconnection link prefixes belongs to both of them, they are also the interfaces addresses on the border routers. .

7. IANA Considerations

TBD.

8. Acknowledgement

The author would like to thank Acee Lindem, Ketan Talaulikar, Jie Dong, Jeff Tantsura and Dhruv Dhody for their valuable comments and suggestions.

9. Normative References

[I-D.ietf-idr-bgp-ls-segment-routing-ext]
Previdi, S., Talaulikar, K., Filsfils, C., Gredler, H., and M. Chen, "BGP Link-State extensions for Segment Routing", draft-ietf-idr-bgp-ls-segment-routing-ext-08 (work in progress), May 2018.

[I-D.ietf-idr-bgpls-segment-routing-epe]
Previdi, S., Filsfils, C., Patel, K., Ray, S., and J. Dong, "BGP-LS extensions for Segment Routing BGP Egress Peer Engineering", draft-ietf-idr-bgpls-segment-routing-epe-15 (work in progress), March 2018.

[I-D.ietf-ospf-ospfv3-lsa-extend]
Lindem, A., Roy, A., Goethals, D., Vallem, V., and F. Baker, "OSPFv3 LSA Extendibility", draft-ietf-ospf-ospfv3-lsa-extend-23 (work in progress), January 2018.

[I-D.ietf-teas-native-ip-scenarios]
Wang, A., Huang, X., Qou, C., Li, Z., Huang, L., and P. Mi, "CCDR Scenario, Simulation and Suggestion", draft-ietf-teas-native-ip-scenarios-01 (work in progress), June 2018.

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, DOI 10.17487/RFC1195, December 1990, <https://www.rfc-editor.org/info/rfc1195>.

[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>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <https://www.rfc-editor.org/info/rfc5305>.

[RFC5316] Chen, M., Zhang, R., and X. Duan, "ISIS Extensions in Support of Inter-Autonomous System (AS) MPLS and GMPLS Traffic Engineering", RFC 5316, DOI 10.17487/RFC5316, December 2008, <https://www.rfc-editor.org/info/rfc5316>.

[RFC5392] Chen, M., Zhang, R., and X. Duan, "OSPF Extensions in Support of Inter-Autonomous System (AS) MPLS and GMPLS Traffic Engineering", RFC 5392, DOI 10.17487/RFC5392, January 2009, <https://www.rfc-editor.org/info/rfc5392>.

[RFC6119] Harrison, J., Berger, J., and M. Bartlett, "IPv6 Traffic Engineering in IS-IS", RFC 6119, DOI 10.17487/RFC6119, February 2011, <https://www.rfc-editor.org/info/rfc6119>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

[RFC7794] Ginsberg, L., Ed., Decraene, B., Previdi, S., Xu, X., and U. Chunduri, "IS-IS Prefix Attributes for Extended IPv4 and IPv6 Reachability", RFC 7794, DOI 10.17487/RFC7794, March 2016, <https://www.rfc-editor.org/info/rfc7794>.

[RFC8362] Lindem, A., Roy, A., Goethals, D., Reddy Vallem, V., and F. Baker, "OSPFv3 Link State Advertisement (LSA) Extensibility", RFC 8362, DOI 10.17487/RFC8362, April 2018, <https://www.rfc-editor.org/info/rfc8362>.

Authors' Addresses

Aijun Wang
China Telecom
Beiqijia Town, Changping District
Beijing, Beijing 102209
China

Email: wangaj.bri@chinatelecom.cn

Huaimo Chen
Huawei Technologies
Boston, MA
USA

Email: Huaimo.chen@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 29, 2020

X. Xu
Alibaba Inc
K. Talaulikar
Cisco Systems
K. Bi
Huawei
J. Tantsura
Apstra
N. Triantafillis
Amazon Web Services
November 26, 2019

BGP Neighbor Discovery
draft-xu-idr-neighbor-autodiscovery-12

Abstract

BGP is being used as the underlay routing protocol in some large-scaled data centers (DCs). Most popular design followed is to do hop-by-hop external BGP (EBGP) session configurations between neighboring routers on a per link basis. The provisioning of BGP neighbors in routers across such a DC brings its own operational complexity.

This document introduces a BGP neighbor discovery mechanism that greatly simplifies BGP operations in such DC and other networks by automatic setup of BGP sessions between neighbor routers using this mechanism.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

BGP is being used as the underlay routing protocol instead of link-state routing protocols like IS-IS and OSPF in some large-scale data centers (DCs). [RFC7938] describes the design, configuration and operational aspects of using BGP in such networks. The most popular design scheme involves the setup of external BGP (EBGP) sessions over individual links between directly connected routers using their interface addresses. Such BGP neighbor provisioning requires configuration of the neighbor IP address and Autonomous System (AS) Number (ASN) for BGP neighbor on each and every link of every BGP router. As a DC fabric comprising of topology described in [RFC7938] grows with addition of new leafs, spines, and links between them, the BGP provisioning needs to be carefully updated. Unlike with the link-state protocols, in the case of BGP, there is no automatic discovery of neighbors and route exchange between them by simply adding links and nodes of the fabric into the routing protocol operation.

In some DC designs with BGP, multiple links are added between a leaf and spine to add additional bandwidth. Use of link-aggregation at Layer 2 level may not be always desirable in such cases due to the risk of flow polarization on account of a mix of ECMP at Layer 2 and Layer 3 levels. In such cases, one option is for EBGP sessions to be setup between two BGP neighbors over each of the links between them. In such a case, the BGP session scale and the resultant increase in update processing may pose scalability challenges. A second option is for a single EBGP session to be setup between the loopback IP addresses between the neighbor and then configure some static routes for loopback reachability over the underlying links. This option introduces an additional provisioning task for the static routes.

Furthermore, there is also a need for BGP to be able to describe its links and its neighbors on its directly connected links and export this information via BGP-LS [RFC7752] to provide a detail link-level topology view of a data center running BGP. The ability of BGP in discovering its neighbors over its links, monitoring their liveliness and learning the link attributes (such as addresses) is required for

the conveying the link-state topology in such a BGP network. This information can be leveraged by the BGP-SPF proposal [I-D.ietf-lsvr-bgp-spf] which introduces link-state routing capabilities in BGP. This information can also be leveraged to convey the link-state topology in a network running traditional BGP routing using BGP-LS as described in [I-D.ketant-idr-bgp-ls-bgp-only-fabric] and to enabled end to end traffic engineering use-cases spanning across DCs and the core/access networks.

1.1. Requirements Language

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.

2. Terminology

This document makes use of the terms defined in [RFC4271] and [RFC7938] .

3. Applicability

The applicability of the BGP Neighbor Discovery mechanism described in this document is limited to deployments where BGP is used as routing protocol between directly connected routers and when there is a requirement for automatic setup of BGP peering between them.

- o In DC networks where BGP is used as a hop-by-hop routing protocol [RFC7938].

- o In metro networks where access aggregation topologies are architected as a CLOS topology (or similar other networks) and BGP is used as a hop-by-hop routing protocol.

While this document uses EBGP examples, the mechanism is equally applicable in designs that use IBGP similarly for hop-by-hop routing.

The applicability of the BGP Neighbor Discovery mechanism to any other BGP protocol deployment is outside the scope of this document.

4. Requirements

This section describe the requirements for the BGP hop-by-hop routing deployments that were considered for the definition of the BGP Neighbor Discovery extensions proposed in this document..

Following are the key requirements related for the BGP neighbor discovery process:

1. It should perform discovery of directly connected BGP routers. Mechanism should support either IPv4 or IPv6 or a dual stack design and it should be generic for any link-layer.

2. It should include exchange of BGP peering addresses (IPv4 or IPv6 or both) that routers can use to automatically setup BGP TCP peering between themselves. The mechanism should leverage the existing capability negotiation process performed as part of the BGP TCP session establishment.

3. When BGP peering is desired to be performed over loopback addresses of the routers, then the mechanism should automatically setup reachability to the loopback over one or more underlying directly connected links between them. In this scenario, the mechanism should also provide resolution for the BGP next-hop address (i.e. the loopback address) for the BGP routes exchanged over these sessions between the loopback addresses.

4. Mechanism should enable exchange of link-level information such as IP addresses and link attributes between the directly connected BGP routers. It should be extensible to include other information in the future.

5. Mechanism should be limited to link scope for security and use link-local addressing only. Cryptographic mechanisms should be also provided for additional security.

6. Mechanism should support capabilities for performing optional validation of parameters to detect misconfiguration (e.g. link address subnet mismatch, peering between incorrect AS, etc.) in an extensible manner before going on to use the link and the setup of the BGP TCP peering session over it.

7. The mechanism should not affect or change the BGP TCP session establishment procedures and the BGP routing exchange over the TCP session other than the interactions for triggering the setup/ removal of peer session that is based on discovery mechanism.

8. The mechanism should leverage existing fast-detection techniques for failures that are used currently for EBGP sessions over directly connected links like fast-external-failover and BFD.

9. The mechanism should focus on the discovery process and exchange of status as a control plane procedure and be sufficiently loosely coupled with the base BGP operations to enable

implementations to ensure scalability of BGP operations when using the discovery procedures.

5. Overview

At a high level, this specification introduces the use of UDP based BGP Hello messages to be exchanged between directly connected BGP routers for neighbor discovery.

1. Information is exchanged between BGP routers on a per link basis leading to discovery of each others peering address and other information.

2. The TCP session establishment for the BGP protocol operation and the BGP routing exchange over these sessions can then follow without any change/modification from the existing BGP protocol operations as specified in [RFC4271].

3. As part of the neighbor information exchange the route to a neighbor's peering address is also automatically setup pointing over the links over which the neighbor is discovered.

4. This route is used for both the BGP TCP session establishment as well as for resolution of the BGP next-hop (NH) for the routes learnt via the neighbor instead of an underlying IGP or static route.

This document prefers the use of an extension to BGP protocol since the deployments and use-cases targeted (i.e. large-scale DCs) are already running BGP as their routing protocol. Extending BGP with neighbor discovery capabilities is operationally and implementation wise a simpler approach than requiring a new or an additional protocol to be first extended to do this functionality (to exchange BGP-specific parameters) and then also integrated its operations with BGP protocol operations.

The BGP Neighbor discovery mechanism is a control plane mechanism intended to discovery and maintain the BGP router's adjacencies with its neighbors over directly connected links. Maintaining an adjacency also involves detecting any changes in parameters using periodic messages and triggering corresponding actions based on the change. Such actions also include removal of the BGP TCP peering for an auto discovered peering session based on the neighbor discovery. However, the mechanism is not intended for a fast liveness detection of neighbor and existing mechanisms for this purpose such as BFD [RFC5880] may be leveraged.

The BGP Neighbor discovery mechanism is scoped to a link and works using link-local addressing. In a BGP DC network that is using IPv6 in the fabric underlay, it is possible that no IPv6 global addresses are assigned to the interfaces between the nodes and the IPv6 Global address(es) are assigned only to the loopback interfaces of these nodes. The Neighbor discovery mechanism enables the setup of BGP peering using the IPv6 Global addresses on the loopback interfaces and hop by hop routing with just IPv6 link-local addresses on the interfaces. Such a design eases introduction of nodes in the fabric and links between them from a provisioning aspect. In a deployment with IPv4 addressing, IP unnumbered could be similarly used for all the links between the nodes using the IPv4 address assigned to the loopback interfaces on those nodes.

The BGP neighbor discovery mechanism defined in this document borrows ideas from the Label Distribution Protocol (LDP) [RFC5036]. However, most importantly, only the concept of link-local signaling based neighbor discovery is borrowed while the discovery aspect for targeted LDP sessions does not apply to this BGP neighbor discovery mechanism.

The further sections in this document first describe the newly introduced message formats and TLVs and then go on to describe the procedures of BGP neighbor discovery and its integration with the base BGP protocol mechanism as specified in [RFC4271].

The operational and management aspects of the BGP neighbor discovery mechanism are described in Section 12.

6. UDP Message Header

The BGP neighbor discovery mechanism will operate using UDP messages. The UDP port of TBD (179 is the preferred port number to be assigned as specified in Section 13) is used which is same as the TCP port 179 used by BGP. The BGP UDP message common header format is specified as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Version   |     Type      |      Message Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           AS number                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         BGP Identifier                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: BGP UDP Message Header

Version: This 1-octet unsigned integer indicates the protocol version number of the message. The current BGP version number is 4.

Type: The type of BGP message

Message Length: This 2-octet unsigned integer specifies the length in octets of the entire BGP UDP message including the header.

AS number: AS Number of the UDP message sender.

BGP Identifier: BGP Identifier of the UDP message sender.

BGP UDP messages can be sent using either IPv4 or IPv6 depending on the address used for session establishment and provisioned on the interfaces over which these messages are sent.

7. Hello Message Format

A BGP router uses UDP based Hello messages to discover directly connected BGP neighbors over those interfaces enabled for Neighbor Discovery. The BGP Hello messages for the Neighbor Discovery procedure are used for link-locally signaling and hence MUST be addressed to the "all routers on this subnet" group multicast address (i.e., 224.0.0.2 in the IPv4 case and FF02::2 in the IPv6 case) and the TTL for the IP packets SHOULD be set to 1. The IP source address MUST be set to the address of the interface over which the message is sent out which would be the primary interface address or unnumbered address in the IPv4 case and the IPv6 link-local address on the interface in the IPv6 case.

The Hello message format is as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Version   |     Type      |      Message Length           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           AS number                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         BGP Identifier                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Adjacency Hold Time       |    Flags      |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             TLVs                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: BGP Hello Message

Version: This 1-octet unsigned integer indicates the protocol version number of the message. The current BGP version number is 4.

Type: The type of BGP message (Hello - TBD value from BGP Message Types Registry)

Message Length: This 2-octet unsigned integer specifies the length in octets of the TLVs field.

AS number: AS Number of the BGP router sending the Hello message.

BGP Identifier: BGP Identifier of the BGP router sending the Hello message.

Adjacency Hold Time: Hello adjacency hold timer in seconds. Adjacency Hold Time specifies the time, for which the receiving BGP neighbor router SHOULD maintain adjacency state for it, without receipt of another Hello. A value of 0 means that the receiving BGP peer should immediately mark that the adjacency to the sender is going down.

Flags : Current defined bits are as follows. All other bits SHOULD be cleared by sender and MUST be ignored by receiver.

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|S|               |
+-+-+-+-+-+-+-+-+
```

where:

S bit - indicates that this is a State Change Hello message when SET and normal periodic Hello message when CLEAR

Reserved: SHOULD be set to 0 by sender and MUST be ignored by receiver.

TLVs: This field contains one or more TLVs as described below.

BGP HELLO messages can be sent using either IPv4 or IPv6 addresses depending on the addressing used for session establishment and provisioned on the interfaces over which these messages are sent. When both IPv4 and IPv6 is enabled on the interface, then IPv6 address SHOULD be used. Implementations MAY provide an option to override the choice of address family to be used. The choice of address family to be used MUST be consistent on all BGP routers on a given link for neighbor discovery.

Based on the setting of the S flag, there are two variants of the Hello message:

1. State Change Hello Message : these Hello messages include TLVs which convey the state and parameters of the local interface and adjacency to other routers on the link. They are generated only when there is a change in state of the adjacency or some parameter at the interface level.

2. Periodic Hello Message : these are the normal periodic Hello messages which do not include TLVs and are used to maintain the adjacency on the link during steady state conditions.

These Hello message variants are intended to limit the exchange of information and state via TLVs to only those periods where necessary while using lightweight Hello messages during steady state. This simplifies the Hello message processing and improves scalability of the discovery mechanism.

The neighbor discovery procedure using the Hello message is described in Section 9 and its relation with the BGP Keepalives and Hold Timer for the TCP session is described in Section 10.

8. Hello Message TLVs

The BGP Hello message carries TLVs as described in this section that enable exchange of information on a per interface basis between directly connected BGP neighbors. These messages enable the neighbor discovery process.

8.1. Accepted ASN List TLV

The Accepted ASN List TLV is an optional TLV that is used to signal an unordered list of AS numbers from which the BGP router would accept BGP sessions. When not signaled, it indicates that the router will accept BGP peering from any ASN from its neighbors. Indicating the list of ASNs, helps avoid the neighbor discovery process getting stuck in a 1-way state where one side keeps attempting to setup adjacency while the other does not accept it due to incorrect ASN.

The operational and management aspects of this ASN based policy control for BGP neighbor discovery are described further in Section 12.

This TLV SHOULD NOT be included in a Hello message with the S bit CLEAR. More than a single instance of this TLV MUST NOT be included in a Hello message. If a router receives multiple instances of this

TLV then it should only consider the first instance in the sequence and ignore the rest.

The format of this TLV is shown below

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Type                 |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Accepted ASN List(variable)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Accepted ASN List TLV

Type: TBD1

Length: Specifies the length of the Value field in octets (in multiple of 4)

Accepted ASN-List: This variable-length field contains one or more accepted 4-octet ASNs.

8.2. Peering Address TLV

The Peering Address TLV is used to indicate to the neighbor the address to be used for setting up the BGP TCP session. Along with the peering address, the router can specify its supported AFI/SAFI(s). When the AFI/SAFI values are specified as 0/0, then it indicates that the neighbor can attempt for negotiation of any AFI/SAFIs. The indication of AFI/SAFI(s) in the Peering Address TLV is not intended as an alternative for the MP capabilities negotiation mechanism done as part of the BGP TCP session establishment.

Multiple instances of this TLV MAY be included in the Hello message, one for each peering address (e.g. IPv4 and IPv6 or multiple IPv4 addresses for different AFI/SAFI sessions). When multiple peering addresses are provisioned, then the indication helps the router select the appropriate peer address of the neighbor based on its local peering address profile by matching the supported AFI/SAFIs.

This TLV is essential for the setting up of the TCP peering between BGP neighbors using the neighbor discovery mechanism. When a BGP router stops including a Peer Address in its State Change Hello messages, then it is no longer accepting TCP peering sessions to that address and the neighbor SHOULD clean up any peering session that was setup to that address via the discovery mechanism.

Implementations SHOULD support the signaling of an interface IP address in the Peering Address TLV and perform the BGP TCP session establishment using interface addresses (i.e. the neighbor discovery mechanism is not limited to the use of loopback addresses for the peering session establishment). Implementations MAY support the signaling of IPv6 Link Local addresses using the Peering Address TLV and using the same for the BGP TCP session setup.

This TLV SHOULD NOT be included in a Hello message with the S bit CLEAR.

The Peering Address TLV format is shown below.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Type                 |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Flags       | No. AFI/SAFI  |      Reserved                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Address (4-octet or 16-octet)                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            AFI                |   SAFI        |  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      sub-TLVs ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Peering Address TLV

Type: TBD2

Length: Specifies the length of the Value field in octets.

Flags : Current defined bits are as follows. All other bits SHOULD be cleared by sender and MUST be ignored by receiver.

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|A|             |
+-+-+-+-+-+-+-+-+
```

where:

A bit - address is IPv6 when SET and IPv4 when CLEAR

Number of AFI/SAFI: indicates the number of AFI/SAFI pairs that the router supports on the given peering address.

Reserved: sender SHOULD set to 0 and receiver MUST ignore.

Address: This 4 or 16 octet field indicates the IPv4 or IPv6 address which is used for establishing BGP sessions.

AFI/SAFI : one or more pairs of these values that indicate the supported capabilities on the peering address.

Sub-TLVs : optional and currently none defined

8.3. Local Prefix TLV

BGP neighbor discovery mechanism, in certain scenarios, requires a BGP router to program a route in its local routing table for a prefix belonging to its neighbor router. On such scenario is when the BGP TCP peering is to be setup between the loopback addresses on the neighboring routers. This requires that the routers have reachability to their each other’s loopback addresses before the TCP session can be brought up.

The Local Prefix TLV is an optional TLV which enables a BGP router to explicitly signal its local prefix to its neighbor for setting up of such a local routing entry pointing over the underlying link over which it is being signaled. This enables the BGP router to have control over the specific links over which its neighbor that may reach it for the specific local prefix. The details of the procedure for programming of the route corresponding to the prefix signaled using the Local Prefix TLV is described in Section 9.3..

Multiple instances of the Local Prefix TLV MAY be included in the Hello message with each carrying a specific prefix in it. This TLV SHOULD NOT be included in a Hello message with the S bit CLEAR.

The Local Prefix TLV format is as shown below.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Type                 |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Flags       | Prefix Length |      Reserved                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Prefix Address (4-octet or 16-octet)                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  sub-TLVs ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Local Prefix TLV

Type: TBD3

Length: Specifies the length of the Value field in octets

Flags : Current defined bits are as follows. All other bits SHOULD be cleared by sender and MUST be ignored by receiver.

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|A|             |
+-+-+-+-+-+-+-+-+
```

where:

A bit - address is IPv6 when SET and IPv4 when CLEAR

Prefix Length: specifies the Prefix length

Reserved: sender SHOULD set to 0 and receiver MUST ignore.

Prefix Address: This 4 or 16 octet field indicates the IPv4 or IPv6 prefix address.

Sub-TLVs : optional and currently none defined

8.4. Link Attributes TLV

The Link Attributes TLV is a mandatory TLV in a State Change Hello message that signals to the neighbor the link attributes of the interface on the local router. One and only one instance of this TLV MUST be included in the State Change Hello message. A State Change Hello message without this TLV included MUST be discarded and an error logged for the same.

This TLV enables a BGP router to learn all its neighbors IP addresses on the specific link as well as it's link identifier. When the interface is IPv4 enabled, all the IPv4 addresses configured on it are included in this TLV. IPv4 unnumbered address is not included in this TLV and no IPv4 address would be included for the interface in such cases. When the interface is IPv6 enabled, all the IPv6 global addresses configured on the interface are included in this TLV. IPv6 link-local addresses are not included in this TLV. In case of an interface running dual stack, both IPv4 and IPv6 addresses are included in this TLV irrespective of the address family that is used for UDP message exchange.

Additional sub-TLVs may be defined in the future to exchange other link attributes between BGP neighbors. This TLV SHOULD NOT be included in a Hello message with the S bit CLEAR.

The Link Attributes TLV format is as shown below.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Type                 |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Local Interface ID       |      Flags    |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    No. of IPv4 Addresses      |      No. of IPv6 Addresses    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   IPv4 Interface Address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Prefix Mask  | ...
+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                IPv6 Global Interface Address                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Prefix Mask  | ...
+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  sub-TLVs ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Link Attributes TLV

Type: TBD4

Length: Specifies the length of the Value field in octets

Local Interface ID : the local interface ID of the interface (refer unnumbered link section of [RFC2104] e.g. the MIB-2 ifIndex). This helps uniquely identify the link even when there are multiple links between two neighbors using IPv4 unnumbered address or only having IPv6 link-local addresses.

Flags : Currently defined bits are as follows. Other bits SHOULD be cleared by sender and MUST be ignored by receiver.

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|I|V|B|         |
+-+-+-+-+-+-+-+-+
```

where:

I bit - indicates link is enabled for IPv4

V bit - indicates link is enabled for IPv6

B bit - indicates support for BFD monitoring [RFC5880] over the link

Reserved: SHOULD be set to 0 by sender and MUST be ignored by receiver.

No. of IPv4 Addresses : specifies the number of IPv4 addresses on the interface. When value is 0, then it indicates no IPv4 Prefixes are present or the interface is IPv4 unnumbered if it is enabled for IPv4

No. of IPv6 Addresses : specifies the number of IPv6 global addresses on the interface. When value is 0, then it indicates no IPv6 Global Prefixes are present and the interface is only configured with IPv6 link-local addresses if it is enabled for IPv6.

IPv4 Address & Mask: Zero or more pairs of IPv4 address and their mask.

IPv6 Address & Mask: Zero or more pairs of IPv6 address and their mask.

Sub-TLVs : optional and currently none defined

8.5. Neighbor TLV

The Neighbor TLV is used by a BGP router to indicate its Hello adjacency state with its neighboring router(s) on the specific link. The neighbor is identified by its AS Number and BGP Identifier. The router MUST include the Neighbor TLV for each of its discovered neighbors on that link irrespective of its status.

The usage of the Neighbor TLV is described in detail in Section 9. This TLV SHOULD NOT be included in a Hello message with the S bit CLEAR.

The Neighbor TLV format is as shown below.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Type                 |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Flags       |   State       |      Reserved                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Neighbor AS number                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Neighbor BGP Identifier                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  sub-TLVs ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Neighbor TLV

Type: TBD5

Length: Specifies the length of the Value field in octets

Flags : Current defined bits are as follows. All other bits SHOULD be cleared by sender and MUST be ignored by receiver.

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|B|               |
+-+-+-+-+-+-+-+-+
```

where:

B bit - When SET with the adjacency state not in Accepted state indicates that the adjacency is not accepted due to BFD down.

State : Indicates the state code of the adjacency state machine (refer to Section 9.2 for details) for the neighbor over this link. The following codes are currently defined

0 - Down (not to be used as state in this TLV

1 - Initial (not to be used as state in this TLV)

2 - 1-way

3 - 2-way

4 - Adj-Reject

5 - Adj-OK

6 - Accepted

Reserved: SHOULD be set to 0 by sender and MUST be ignored by receiver.

Neighbor AS number: AS Number of the neighbor BGP router as signaled in its Hello message.

Neighbor BGP Identifier: BGP Identifier of the neighbor BGP router as signaled in its Hello message.

Sub-TLVs : currently none defined

8.6. Cryptographic Authentication TLV

The Cryptographic Authentication TLV is an optional TLV that is used as part of an authentication mechanism for BGP Hello message by securing against spoofing attacks. It also introduces a cryptographic sequence number carried in the Hello messages that can be used to protect against replay attacks. Using this Cryptographic Authentication TLV, one or more secret keys (with corresponding Security Association (SA) IDs) are configured on each BGP router. For each BGP Hello message, the key is used to generate and verify an HMAC Hash that is stored in the Cryptographic Authentication TLV. For the cryptographic hash function, this document proposes to use SHA-1, SHA-256, SHA-384, and SHA-512 defined in US NIST Secure Hash Standard (SHS) [FIPS-180-4]. The HMAC authentication mode defined in [RFC2104] is used. Of the above, implementations MUST include support for at least HMAC-SHA-256, SHOULD include support for HMAC-SHA-1, and MAY include support for HMAC-SHA-384 and HMAC-SHA-512.

Further details for ensuring the security of the BGP Hello UDP messages are described in Section 11.

The Cryptographic Authentication TLV format is as shown below.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Type                 |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Security Association ID                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Cryptographic Sequence Number (High-Order 32 Bits)      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Cryptographic Sequence Number (Low-Order 32 Bits)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Authentication Data (Variable)                //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: Cryptographic Authentication TLV

Type: TBD6

Length: Specifies the length of the Value field in octets

Security Association ID: The 32-bit field that maps to the authentication algorithm and the secret key used to create the message digest carried in Hello message payload.

Cryptographic Sequence Number: The 64-bit, strictly increasing sequence number that is used to guard against replay attacks. The 64-bit sequence number MUST be incremented for every BGP Hello message sent by the BGP router. Upon reception, the sequence number MUST be greater than the sequence number in the last BGP Hello message accepted from the sending BGP neighbor. Otherwise, the BGP hello message is considered a replayed packet and is dropped. The Cryptographic Sequence Number is a single space per BGP router.

Authentication Data: This field carries the digest computed by the Cryptographic Authentication algorithm in use. The length of the Authentication Data varies based on the cryptographic algorithm in use, which is shown below:

HMAC-SHA1 20 bytes

HMAC-SHA-256 32 bytes

HMAC-SHA-384 48 bytes

HMAC-SHA-512 64 bytes

9. Neighbor Discovery Procedure

The neighbor discovery mechanism in BGP is implemented with the introduction of an Interface state in BGP and an Adjacency Finite State Machine (FSM). This section describes the states, FSM and procedures involved.

9.1. Interface Procedures

In order to perform neighbor discovery, BGP needs to maintain state for the subset of its connected interfaces over which neighbor discovery is enabled. For these interfaces, BGP sends its Hello messages, including the TLVs described in Section 8, as long as its link is UP. The Neighbor TLV described in Section 8.5 is, included once a neighbor is discovered as described in Section 9.2 .

The Hello messages MUST be originated periodically at an interval which is less than or equal to one third of the Adjacency Hold Time indicated by the router in its Hello message. The RECOMMENDED default value for the Adjacency Hold Time is 45 seconds which makes the hello message interval to be 15 seconds. Period Hello messages ensure robustness of the neighbor discovery mechanism against transient loss of hello messages that are sent over unreliable UDP messaging channel and also enable detection of neighbor down events over specific links. Periodic Hello messages that do not convey any change in state SHOULD exclude TLVs that signal the local interface or adjacency state and have the S bit CLEAR as specified in Section 7.

A State Change Hello message MUST be triggered, without waiting for the periodic timer expiry, whenever there is a change in the router's Hello TLVs' content that needs to be signaled to its neighbor over the specific link. A State Change Hello message MUST also be triggered when a new neighbor's Hello message is first received or change is detected in the neighbor's Hello TLV's that results in change in it's adjacency state. Once a State Change Hello message is triggered on a specific interface, the router MUST continue to generate State Change Hello messages on it with the necessary TLVs included at periodic hello message intervals for a period of time that is at least equal to the Adjacency Hold Time. This ensures that messages carrying the updated information and local state changes are not lost. The router can switch back to Periodic Hello messages

after it has transmitted State Change Hello messages with the latest TLV contents for the Adjacency Hold Time period.

When a router receives a Hello message from its neighbor, it MUST restart the Adjacency Hold timer that it is maintaining for the neighbor adjacency using the value indicated in the Hello message. When the message is of type State Change (i.e. with S bit SET), it additionally needs to process all the TLVs included and verify the signaled state against what was conveyed in the previous State Change Hello message from the same neighbor. Any changed identified would trigger the adjacency FSM change as described in Section 9.2.

When a router does not receive a Hello message from its neighbor for a period equal to Adjacency Hold Time, then it MUST treat this as an adjacency down event and clean up its adjacency state to this neighbor as described in Section 9.2.

Before the interface is shut or the neighbor discovery mechanism is disabled on it, the router SHOULD attempt to send out immediate Hello messages, with the S bit CLEAR (i.e. not including state related TLVs) and with Adjacency Hold Time set to 0, to trigger the adjacency down event on its neighbors. It MUST then clean up its own adjacency states on that specific link.

When either the BGP Identifier or the AS number are modified, then the router MUST send out a triggered Hello message, with the S bit CLEAR and with Adjacency Hold Time set to 0 using the old BGP Identifier and AS number values, over all the links enabled for BGP neighbor discovery.

A router receiving a Hello message with Adjacency Hold Time set to 0 MUST treat this event as if the adjacency hold timer has expired for the specific neighbor and proceed to bring down the adjacency.

An interface going down (e.g. due to link failure or loss of signal) MUST immediately trigger the adjacency down event for all adjacencies over it as if the adjacency hold timer expired for all neighbors on that link.

9.2. Adjacency State Machine

On a per interface basis, BGP needs to maintain an adjacency state for each neighbor that it discovers. The adjacency state is maintained as a FSM and it has states as described in the following sections.

9.2.1. Down State

This is the transient terminal state after which an adjacency is deleted.

When transitioning to the Down state from Accepted, the router removes the path corresponding to this adjacency from any Adjacency Route that it had setup to the neighbor's prefixes. If no other adjacency exists in Accepted state to the neighbor, then it also deletes the BGP TCP peering session(s) setup to the neighbor based on the neighbor discovery mechanism.

9.2.2. Initial State

This is the transient initial state from which an adjacency starts, when the router detects a hello message from a new neighbor on the link, and immediately transitions to the 1-way state.

9.2.3. 1-Way State

While in the 1-way state (or when entering it), the adjacency transitions from 1-way to 2-way state when the router detects a Neighbor TLV corresponding to itself in the neighbor's Hello message. If the state does not immediately transition on to 2-way after entering 1-way, the the router MUST immediately trigger a State Change Hello message with the inclusion of the neighbor in a Neighbor TLV with the state set to 1-way.

When transitioning to the 1-way state from Accepted, the router removes the path corresponding to this adjacency from any Adjacency Route that it had setup to the neighbor's prefixes. If no other adjacency exists in Accepted state to the neighbor, then it also deletes the BGP TCP peering session(s) setup to the neighbor based on the neighbor discovery mechanism.

Adjacency transitions to Down state for any of the following events:

- Link goes down operationally or is administratively shut
- Adjacency Hold Timer expires
- Router receives a Hello message from its neighbor with Adjacency Hold Time value set to 0
- Neighbor discovery is disabled on the link
- Change in BGP Identifier or AS number on the local router

9.2.4. 2-Way State

Upon transitioning into this state, the router triggers a State Change Hello message with the neighbor's status set to 2-way in the Neighbor TLV. At this stage, both neighbors have received each other's Hello messages and thus discovered each other.

When the router, in this adjacency state, detects that the neighbor's state for itself is 2-way or higher, then it performs the validation checks based on local policy and information exchanged in the Hello TLVs. Following are some of the validation checks that may be performed on the adjacency:

- o Verify subnet matching between the local and remote interface addresses.

- o Verify AS numbers based on local policy as well as against the Allowed ASN TLV when one is being exchanged.

- o Verify that BFD monitoring (when enabled) is indicating UP state.

When the adjacency passes the validation checks, it transitions to the Adj-OK state and transitions to the Adj-Reject state otherwise.

The adjacency transitions to Down state for any of the adjacency down events described in Section 9.2.3 .

The adjacency transitions to 1-way state when the router stops seeing itself in a Neighbor TLV of its Neighbor's State Change Hello messages.

9.2.5. Adj-Reject State

Upon transitioning into this state, the router triggers a State Change Hello message with the neighbor's status set to Adj-Reject in the Neighbor TLV.

The adjacency remains in the Adj-Reject state as long as the parameters being exchanged via the State Change Hello messages do not pass validation checks. The neighbors continue to include each other in their respective State Change Hello messages.

The adjacency transitions to the Adj-OK state once the validation checks pass (e.g. due to update in any parameters or local policy).

The adjacency transitions to Down state for any of the adjacency down events described in Section 9.2.3 .

The adjacency transitions to 1-way state when the router stops seeing itself in a Neighbor TLV of its Neighbor's State Change Hello messages.

When transitioning to an Adj-Reject state from Accepted state, the router removes the path corresponding to this adjacency from any Adjacency Route that it had setup to the neighbor's prefixes. If no other adjacency exists in Accepted state to the neighbor, then it also deletes the BGP TCP peering session(s) setup to the neighbor based on the neighbor discovery mechanism.

9.2.6. Adj-OK State

Upon transitioning into this state, the router triggers a State Change Hello message with the neighbor's status set to Adj-OK in the Neighbor TLV.

The adjacency transition to Adj-OK state indicates that the router has accepted its neighbor. However, it is possible that the neighbor has not accept it and is signaling Adj-Reject state for the adjacency from it's end.

The adjacency transitions to the Accepted state from Adj-OK once it detects that its neighbor is also signaling the Adj-OK or Accepted state for it.

The adjacency transitions to Down state for any of the adjacency down events described in Section 9.2.3 .

The adjacency transitions to 1-way state when the router stops seeing itself in a Neighbor TLV of its Neighbor's State Change Hello messages.

The adjacency transitions to Adj-Reject state when any of the validation checks listed in Section 9.2.4 fail.

When transitioning to an Adj-OK state from Accepted state, the router removes the path corresponding to this adjacency from any Adjacency Route that it had setup to the neighbor's prefixes. If no other adjacency exists in Accepted state to the neighbor, then it also deletes the BGP TCP peering session(s) setup to the neighbor based on the neighbor discovery mechanism.

9.2.7. Accepted State

The adjacency transition to Accepted state indicates that both the neighboring routers have accepted the adjacency to each other.

On this transition, the router triggers a State Change Hello message with the neighbor's status set to Accepted in the Neighbor TLV. It then installs the Adjacency Route(s) for the Prefix(es) signaled by the neighbor via the Local Prefix TLV via this adjacency link using the neighbor's address on that link. If this is the first Accepted adjacency to the neighbor then the Adjacency Route gets added to the local routing table, otherwise an additional path corresponding to this adjacency link and neighbor address on it gets added to the existing Adjacency Route. The details are described in Section 9.3.

When this is the first Accepted adjacency to the neighbor, then the setup of the BGP TCP session to the Peering Address(es) signaled by the neighbor is also triggered.

The adjacency transitions to Down state for any of the adjacency down events described in Section 9.2.3.

The adjacency transitions to 1-way state when the router stops seeing itself in a Neighbor TLV of its Neighbor's State Change Hello messages.

The adjacency transitions to Adj-Reject state when any of the validation checks listed in Section 9.2.4 fail.

9.3. Adjacency Route

The Adjacency Route programming is an optional part of the BGP Neighbor Discovery mechanism for setting up reachability for the neighbor's prefixes signaled via the Local Prefix TLV corresponding to adjacencies in Accepted state.

Adjacency Routes establish reachability between local prefixes on directly connected BGP routers. They enable reachability between the Peering Addresses (generally loopbacks) of the two neighbors so that the BGP TCP session may come up between them. Then, for the BGP routes learnt over the TCP session, where the next-hop is the neighbor, they also provide the BGP NH resolution.

Unlike other BGP routes, these are not recursive routes as in they point to the neighbor's interface and IP address. These routes that are setup as part of the neighbor discovery procedure are hence different from the regular IBGP and EBGP routes. These routes also MUST have a better administrative distance as compared to the IBGP and EBGP routes to ensure that they do not get displaced from the forwarding by BGP routes learnt over the very session(s) established using these peering routes.

The Adjacency Routes SHOULD NOT be stored in any of BGP RIBs [RFC4271] since they are not computed based on the BGP decision process. It is RECOMMENDED that these routes be managed in a separate routing table within the BGP Neighbor Discovery function to ensure that none of the processing and validation for BGP RIB affects them and in turn they do not influence the BGP decision process and route calculation.

When there are multiple interconnecting links between two BGP neighbors, a single BGP TCP session may be setup between them over which routes are then exchanged. However, in the forwarding, the Adjacency route will have multiple paths - one for each of these interconnecting links. So the BGP routes learnt over the session actually end up getting resolved over this Adjacency route and in turn gets the ECMP load balancing even with a single BGP session.

10. Interactions with Base BGP Protocol

The BGP Finite State Machine (FSM) as specified in [RFC4271] is unchanged and the BGP TCP session establishment, route updates and processing continues to follow the BGP protocol specifications.

BGP peering addresses along with their respective ASNs have traditionally been explicitly provisioned on both BGP neighbors. The difference that neighbor discovery mechanism brings about is in elimination of this configuration as these parameters are learnt via the neighbor discovery procedure. Once BGP router learns its neighbor’s peering address and ASN, then its initializes the BGP Peer FSM for this neighbor in the Idle State - just as if this neighbor was configured. From thereon, the BGP Peer FSM actions follows.

The BGP Keepalives and Hold Timer for the session over TCP apply unchanged and they govern the operations of the BGP TCP session. While the BGP Keepalive works at the TCP session level, the BGP Adjacency Hold Timer monitors one or more underlying interconnecting link adjacencies between the neighbors. The reachability for the BGP TCP session may also be over the some BGP routes learnt via routing updates over the sessions setup via neighbor discovery. It is likely that even after all the underlying interconnecting link adjacencies between two neighbors are down that the neighbor’s peering address is reachable via BGP routing over some other path in the network. In order to avoid this, it is RECOMMENDED that the BGP TCP sessions setup via neighbor discovery mechanism use TTL set to 1 to ensure they are setup only over directly attached links to the neighbors.

Since the BGP TCP session setup via neighbor discovery was meant for hop-by-hop routing, it would be necessary to bring down the session even while its BGP Hold Timer has not expired for faster convergence.

Therefore, when all the underlying link adjacencies between two BGP neighbors move out of the Accepted state (or go down), then the BGP TCP peering session that was setup using BGP Neighbor Discovery mechanism between these two neighbors is also deleted as if it was un-configured.

Since the BGP neighbor discovery mechanism runs over a UDP socket, it is isolated from the core BGP protocol working which is TCP based. Implementations SHOULD ensure that the hello processing does not affect the base BGP operations and scalability. One option may be to run the BGP neighbor discovery mechanism in a separate thread from the rest of BGP processing. These implementation details, however, are outside the scope of this document.

It is not generally expected that BGP sessions are explicitly provisioned along with the neighbor discovery mechanism. However, in such an event, the neighbor discovery mechanism MUST NOT affect or result in any changes to provisioned BGP neighbors and their operations. Specifically, BGP peering to auto-discovered neighbors MUST NOT be instantiated using the procedures described in this document when the same BGP neighbor is already provisioned. The configured BGP neighbor parameters take precedence and the auto-discovered values and parameters are not used for such configured BGP sessions.

11. Security Considerations

BGP routers accept TCP connection attempts to port 179 only from the provisioned BGP neighbors or, in some implementations, those from within a configured address range. With the BGP neighbor auto-discovery mechanism, it is now possible for BGP to automatically learn neighbors and initiate/receive TCP connections from them. This introduces the need for specific considerations to be taken care of to ensure security of the BGP protocol operations.

This document introduces UDP messages in BGP for the neighbor discovery mechanism using the BGP Hello messages. For security purposes, implementations MUST exchange the Hello messages only on interfaces specifically enabled for neighbor discovery. Hello messages MUST NOT be accepted on other than the 224.0.0.2 or FF02::2 addresses. Optionally, implementations MAY set TTL to 255 when originating the Hello messages and receivers check specifically for the TLV to be 254 and discard the packet when this is not the case. This ensures that the Hello packets signaling happens between directly connected BGP routers only.

The BGP neighbor discovery mechanism is expected to be run typically in DCs and between physically connected routers that are trustworthy.

The Cryptographic Authentication TLV (as described in Section 8.6) SHOULD be used in deployments where this assumption of trustworthiness is not valid. This mechanism is similar to one defined for LDP Hello messages that are also UDP based as specified in [RFC7349]. An updated future version of this document will describe similar procedures for BGP hello in more details.

Once the BGP hello messages and the neighbor discovery mechanism is secured, then the security considerations for BGP protocol operations apply for the auto-discovered neighbor sessions.

12. Manageability Considerations

This section is structured as recommended in [RFC5706].

12.1. Operational Considerations

The BGP neighbor discovery mechanism introduced by this document is not applicable to general BGP deployments as discussed in Section 3. The mechanism is specifically meant for networks where BGP is used as a hop-by-hop routing protocol E.g. as described in [RFC7938]. The neighbor discovery mechanism hence SHOULD NOT be enabled by default in BGP.

Implementations SHOULD provide configuration methods that allow enablement of BGP neighbor discovery on specific local interfaces. In a DC network, it is expected that the operator selects the appropriate links on which to enable this e.g. on a Tier 2 node it is enabled on all links towards the Tier 1 and Tier 3 nodes while on a Tier 1 node, it may be only enabled on the links towards the Tier 2 node. The details of this enablement are outside the scope of this document since it varies based on the DC design and may be implementation specific.

Implementations SHOULD provide configuration methods that enable the setup of BGP neighbor templates that enables operator to setup BGP neighbor discovery parameters on the BGP router. Some of the aspects to be considered in such a template are:

- Local address to be used for the BGP TCP session peering along with the local ASN and the AFI/SAFI enabled for the auto-discovered sessions

- BGP policies to be enabled for the auto-discovered sessions

- Optionally specify the list of ASNs with which auto-discovered sessions should be brought up. This is to ensure that when links between different Tier nodes are not used by BGP when they get

connected wrongly due to accidents (e.g. say a Tier 3 node is connected to a Tier 1 node).

- o Authentication methods that are need to be enabled in an environment which is not secure

- o Local interfaces over which the specific template needs to be applied for BGP neighbor discovery

- o Other parameters like the Adjacency Hold Timer value to be used or other optional features

This mechanism does not impose any restrictions on the way ASNs or addresses are assigned to the nodes. Various automatic provisioning, auto-configuration or zero-touch-provisioning mechanisms may be used.

Implementations SHOULD report the state of the BGP operations over each link enabled for neighbor discovery including the status of all adjacencies learnt over it. Implementations SHOULD also report the operations of the auto-discovered BGP TCP peering sessions similar to the provisioned BGP neighbors.

Implementations SHOULD support logging of events like discovery of an adjacency using neighbor discovery including peering route updates and events like triggering of BGP TCP session establishment for them. Errors and alarms related to loss of adjacencies and tear down of BGP TCP peering sessions SHOULD also be generated so they could be monitored.

12.2. Management Considerations

This document introduces UDP based messaging in BGP protocol and therefore the necessary fault management mechanisms are required to be implemented for the same. Implementations MUST discard unsupported message types or version types other than 4 received over a UDP session. Such messages MUST NOT affect the neighbor discovery mechanism in operation using the Hello messages. Unknown TLVs received via the Hello messages MUST be ignored and the rest of the Hello message MUST be processed. Implementations SHOULD discard Hello messages with malformed TLVs and this should be logged as an error.

13. IANA Considerations

This documents requests IANA for updates to the BGP Parameters registry as described in this section.

13.1. BGP Hello Message

This document requests IANA to allocate a new UDP port (179 is the preferred number) and a BGP message type code for BGP Hello message.

```
 Value   TLV Name                                Reference
 -----   ------------------------------------    -------------
 Service Name: BGP-HELLO
 Transport Protocol(s): UDP
 Assignee: IESG <iesg@ietf.org>
 Contact: IETF Chair <chair@ietf.org>.
 Description: BGP Hello Message.
 Reference: This document -- draft-xu-idr-neighbor-autodiscovery.
 Port Number: 179 (preferred value) -- To be assigned by IANA.
```

13.2. TLVs of BGP Hello Message

This document requests IANA to create a new registry "TLVs of BGP Hello Message" with the following registration procedure:

Registry Name: TLVs of BGP Hello Message.

Value	TLV Name	Reference
0	Reserved	This document
1	Accepted ASN List	This document
2	Peering Address	This document
3	Local Prefix	This document
4	Link Attributes	This document
5	Neighbor	This document
6	Cryptographic Authentication	This document
7-65500	Unassigned	
65501-65534	Experimental	This document
65535	Reserved	This document

14. Acknowledgements

The authors would like to thank Enke Chen, Krishna Swamy and Ramesh Yakkala for their valuable comments and suggestions on this document.

15. Contributors

Satya Mohanty
Cisco
Email: satyamoh@cisco.com

Shunwan Zhuang
Huawei
Email: zhuangshunwan@huawei.com

Chao Huang
Alibaba Inc
Email: jingtan.hc@alibaba-inc.com

Guixin Bao
Alibaba Inc
Email: guixin.bgx@alibaba-inc.com

Jinghui Liu
Ruijie Networks
Email: liujh@ruijie.com.cn

Zhichun Jiang
Tencent
Email: zcjiang@tencent.com

Shaowen Ma
Mellanox
mashaowen@gmail.com

16. References

16.1. 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>.

[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>.

[RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, DOI 10.17487/RFC5036, October 2007, <https://www.rfc-editor.org/info/rfc5036>.

[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>.

16.2. Informative References

[FIPS-180-4]
Technology, N. I. O. S. A., "Secure Hash Standard (SHS), FIPS PUB 180-4", March 2012.

[I-D.ietf-lsvr-bgp-spf]
Patel, K., Lindem, A., Zandi, S., and W. Henderickx, "Shortest Path Routing Extensions for BGP Protocol", draft-ietf-lsvr-bgp-spf-06 (work in progress), September 2019.

[I-D.ketant-idr-bgp-ls-bgp-only-fabric]
Talaulikar, K., Filsfils, C., ananthamurthy, k., Zandi, S., Dawra, G., and M. Durrani, "BGP Link-State Extensions for BGP-only Fabric", draft-ketant-idr-bgp-ls-bgp-only-fabric-03 (work in progress), September 2019.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC5706] Harrington, D., "Guidelines for Considering Operations and Management of New Protocols and Protocol Extensions", RFC 5706, DOI 10.17487/RFC5706, November 2009, <https://www.rfc-editor.org/info/rfc5706>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

[RFC7349] Zheng, L., Chen, M., and M. Bhatia, "LDP Hello Cryptographic Authentication", RFC 7349, DOI 10.17487/RFC7349, August 2014, <https://www.rfc-editor.org/info/rfc7349>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

[RFC7938] Lapukhov, P., Premji, A., and J. Mitchell, Ed., "Use of BGP for Routing in Large-Scale Data Centers", RFC 7938, DOI 10.17487/RFC7938, August 2016, <https://www.rfc-editor.org/info/rfc7938>.

Authors' Addresses

Xiaohu Xu
Alibaba Inc
China

Email: xiaohu.xxh@alibaba-inc.com

Ketan Talaulikar
Cisco Systems
India

Email: ketant@cisco.com

Kunyang Bi
Huawei
China

Email: bikunyang@huawei.com

Jeff Tantsura
Apstra
USA

Email: jefftant.ietf@gmail.com

Nikos Triantafillis
Amazon Web Services
USA

Email: ntriantafillis@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 20, 2021

Y. Zhu
China Telecom
Z. Hu
S. Peng
Huawei Technologies
R. Mwehaire
MTN Uganda Ltd.

November 16, 2020

Signaling Maximum Transmission Unit (MTU) using BGP-LS
draft-zhu-idr-bgp-ls-path-mtu-05

Abstract

BGP Link State (BGP-LS) describes a mechanism by which link-state and TE information can be collected from networks and shared with external components using the BGP routing protocol. The centralized controller (PCE/SDN) completes the service path calculation based on the information transmitted by the BGP-LS and delivers the result to the Path Computation Client (PCC) through the PCEP or BGP protocol.

Segment Routing (SR) leverages the source routing paradigm, which can be directly applied to the MPLS architecture with no change on the forwarding plane and applied to the IPv6 architecture, with a new type of routing header, called SRH. The SR uses the IGP protocol as the control protocol. Compared to the MPLS tunneling technology, the SR does not require additional signaling. Therefore, the SR does not support the negotiation of the Path MTU. Since multiple labels or SRv6 SIDs are pushed in the packets, it is more likely that the packet size exceeds the path mtu of SR tunnel.

This document specifies the extensions to BGP Link State (BGP-LS) to carry maximum transmission unit (MTU) messages of link. The PCE/SDN calculates the Path MTU while completing the service path calculation based on the information transmitted by the BGP-LS.

Requirements Language

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

Status of This Memo

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

Copyright Notice

Table of Contents

1. Introduction

[RFC7752]describes the implementation mechanism of BGP-LS by which link-state and TE information can be collected from networks and shared with external components using the BGP routing protocol [RFC4271]. BGP-LS allows the necessary Link-State Database (LSDB)

and Traffic Engineering Database (TEDB) information to be collected from the IGP within the network, filtered according to configurable policy, and distributed to the PCE as necessary.

The appropriate MTU size guarantees efficient data transmission. If the MTU size is too small and the packet size is large, fragmentation may occur too much and packets are discarded by the QoS queue. If the MTU configuration is too large, packet transmission may be slow. Path MTU is the maximum length of a packet that can pass through a path without fragmentation. [RFC1191] describes a technique for dynamically discovering the maximum transmission unit (MTU) of an arbitrary internet path.

The traditional MPLS tunneling technology has signaling for establishing a path. [RFC3988] defines the mechanism for automatically discovering the Path MTU of LSPs. For a certain FEC, the LSR compares the MTU advertised by all downstream devices with the MTU of the FEC output interface in the local device, and calculates the minimum value for the upstream device.

[RFC3209] specify the mechanism of MTU signaling in RSVP-TE. The ingress node of the RSVP-TE tunnel sends a Path message to the downstream device. The Adspec object in the Path message carries the MTU. Each node along the tunnel receives a Path message, compares the MTU value in the Adspec object with the interface MTU value and MPLS MTU configured on the physical output interface of the local tunnel, obtains the minimum MTU value, and puts it into the newly constructed Path message and continues to send it to the downstream equipment. Thus, the MTU carried in the Path message received by the Egress node is the minimum value of the path MTU. The Egress node brings the negotiated Path MTU back to the Ingress node through the Resv message.

Segment Routing (SR) described in [RFC8402] leverages the source routing paradigm. Segment Routing can be directly applied to the MPLS architecture with no change on the forwarding plane [RFC8660] and applied to the IPv6 architecture with a new type of routing header called the SR header (SRH) [RFC8754]. [I-D.ietf-idr-bgp-ls-segment-routing-ext] defines SR extensions to BGP-LS and specifies the TLVs and sub-TLVs for advertising SR information. Based on the SR information reported by the BGP-LS, the SDN can calculate the end-to-end explicit SR-TE paths or SR Policies.

Nevertheless, Segment Routing is a tunneling technology based on the IGP protocol as the control protocol, and there is no additional signaling for establishing the path. so the Segment Routing tunnel cannot currently support the negotiation mechanism of the MTU. Multiple labels or SRv6 SIDs are pushed in the packets. This causes

the length of the packets encapsulated in the Segment Routing tunnel to increase during packet forwarding. This is more likely to cause packet size exceed the traditional MPLS packet size.

This document specify the extension to BGP Link State (BGP-LS) to carry link maximum transmission unit (MTU) messages.

2. Terminology

This draft refers to the terms defined in [RFC8201], [RFC4821] and [RFC3988].

MTU: Maximum Transmission Unit, the size in bytes of the largest IP packet, including the IP header and payload, that can be transmitted on a link or path. Note that this could more properly be called the IP MTU, to be consistent with how other standards organizations use the acronym MTU.

Link MTU: The Maximum Transmission Unit, i.e., maximum IP packet size in bytes, that can be conveyed in one piece over a link. Be aware that this definition is different from the definition used by other standards organizations.

For IETF documents, link MTU is uniformly defined as the IP MTU over the link. This includes the IP header, but excludes link layer headers and other framing that is not part of IP or the IP payload.

Be aware that other standards organizations generally define link MTU to include the link layer headers.

For the MPLS data plane, this size includes the IP header and data (or other payload) and the label stack but does not include any lower-layer headers. A link may be an interface (such as Ethernet or Packet-over-SONET), a tunnel (such as GRE or IPsec), or an LSP.

Path: The set of links traversed by a packet between a source node and a destination node.

Path MTU, or PMTU: The minimum link MTU of all the links in a path between a source node and a destination node.

3. Deploying scenarios

This document suggests a solution to extension to BGP Link State (BGP-LS) to carry maximum transmission unit (MTU) messages. The MTU information of the link is acquired through the process of collecting link state and TE information by BGP-LS. Concretely, a router maintains one or more databases for storing link-state information about nodes and links in any given area. The router's BGP process can retrieve topology from these IGP, BGP and other sources, and distribute it to a consumer, either directly or via a peer BGP speaker (typically a dedicated Route Reflector). [RFC7176] specifies a possible way of using the ISIS mechanism and extensions for link MTU Sub-TLV. In the case of inter-AS scenario (e.g., BGP EPE), the link MTU of the inter-AS link can be collected via BGP-LS directly.

As per [RFC7752], the collection of link-state and TE information and its distribution to consumers is shown in the following figure.

```
                          +-----------+
                          | Consumer  |
                          +-----------+
                                ^
                                |
                          +-----------+
                          |    BGP    |               +-----------+
                          |  Speaker  |               | Consumer  |
                          +-----------+               +-----------+
                            ^   ^   ^                       ^
                            |   |   |                       |
                  +---------------+   |   +-------------------+   |
                  |                   |                       |   |
            +-----------+       +-----------+             +-----------+
            |    BGP    |       |    BGP    |             |    BGP    |
            |  Speaker  |       |  Speaker  |    . . .    |  Speaker  |
            +-----------+       +-----------+             +-----------+
                  ^                   ^                         ^
                  |                   |                         |
            IGP, BGP, Others    IGP, BGP, Others          IGP, BGP, Others
```

Figure 1: Collection of Link-State and TE Information

Please note that this signaled MTU may be different from the actual MTU, which is usually from configuration mismatches in a control plane and a data plane component.

4. BGP_LS Extensions for Link MTU

[RFC7752] defines the BGP-LS NLRI that can be a Node NLRI, a Link NLRI or a Prefix NLRI. The corresponding BGP-LS attribute is a Node Attribute, a Link Attribute or a Prefix Attribute. [RFC7752] defines the TLVs that map link-state information to BGP-LS NLRI and the BGP-LS attribute. Therefore, according to this document, a new sub-TLV is added to the Link Attribute TLV. It is an independent attribute TLV that can be used for the link NLRI advertised with all the Protocol IDs.

The format of the sub-TLV is as shown below.

```
x  TYPE   - TBD
x  LENGTH - Total length of the value field, it should be 2
x  VALUE  - 2-byte MTU value of the link

                     No. of Octets
+-----------------+
|    MTU value    |       2
+-----------------+
```

Figure 2. Sub-TLV Format for Link MTU

Whenever there is a change in MTU value represented by Link Attribute TLV, BGP-LS should re-originate the respective TLV with the new MTU value.

5. IANA Considerations

This document requests assigning a new code-point from the BGP-LS Link Descriptor and Attribute TLVs registry as specified in section 4.

Value	Description	Reference
TBD	Link MTU	This document

6. Security Considerations

This document does not introduce security issues beyond those discussed in RFC7752.

7. Acknowledgements

8. Contributors

Gang Yan
Huawei
China

Email:yangang@huawei.com

Junda Yao
Huawei
China

Email:yaojunda@huawei.com

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>.

9.2. Informative References

[I-D.ietf-idr-bgp-ls-segment-routing-ext] Previdi, S., Talaulikar, K., Filsfils, C., Gredler, H., and M. Chen, "BGP Link-State extensions for Segment Routing", draft-ietf-idr-bgp-ls-segment-routing-ext-16 (work in progress), June 2019.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/info/rfc1191>.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <https://www.rfc-editor.org/info/rfc3209>.

[RFC3988] Black, B. and K. Kompella, "Maximum Transmission Unit Signalling Extensions for the Label Distribution Protocol", RFC 3988, DOI 10.17487/RFC3988, January 2005, <https://www.rfc-editor.org/info/rfc3988>.

[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>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <https://www.rfc-editor.org/info/rfc4821>.

[RFC7176] Eastlake 3rd, D., Senevirathne, T., Ghanwani, A., Dutt, D., and A. Banerjee, "Transparent Interconnection of Lots of Links (TRILL) Use of IS-IS", RFC 7176, DOI 10.17487/RFC7176, May 2014, <https://www.rfc-editor.org/info/rfc7176>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <https://www.rfc-editor.org/info/rfc8201>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC8660] Bashandy, A., Ed., Filsfils, C., Ed., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with the MPLS Data Plane", RFC 8660, DOI 10.17487/RFC8660, December 2019, <https://www.rfc-editor.org/info/rfc8660>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

Authors' Addresses

Yongqing Zhu
China Telecom
109, West Zhongshan Road, Tianhe District.
Guangzhou 510000
China

Email: zhuyq8@chinatelecom.cn

Zhibo Hu
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: huzhibo@huawei.com

Shuping Peng
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: pengshuping@huawei.com

Robbins Mwehaire
MTN Uganda Ltd.
Uganda

Email: Robbins.Mwehair@mtn.com