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

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

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)
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:

- Prefix sid extensions in the context of SR MPLS, as described in [I-D.ietf-idr-bgp-prefix-sid].
- Egress Peer Engineering (EPE) as described in [I-D.ietf-spring-segment-routing-central-epe].
- BGP signaled SR Policy as described in [I-D.ietf-idr-segment-routing-te-policy].
- Automatic Steering as described in [I-D.ietf-spring-segment-routing-policy] and [I-D.ietf-idr-segment-routing-te-policy].
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.

- 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.

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

- 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.
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:

- Addition of identities 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].

- 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].

- 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.

- 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)
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 automatic-steering
                    |    ++--ro co-flag? enumeration

...

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

...
4. Yang Module

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

module ietf-bgp-sr {
  yang-version 1.1;
  // 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"
}

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";
                }
            }
        }
    }
}
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 {

// 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 {
      }
    }
  }
}
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 leaves 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 Parameters
//
// 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: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:global/bgp:afi-safis/" +
  description
  "Augment BGP SAFI route with steering info";
  uses sr-policy-steering-state;
}

// BGP Signaled SR Policy explicit candidate paths state
augment "/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: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: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/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/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/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/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/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/global/bgp:afi-safis/" +
   description
   "Augment AF route with SRv6 SID info";
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]

[I-D.filsfils-spring-srv6-network-programming]
[I-D.ietf-6man-segment-routing-header]

[I-D.ietf-idr-bgp-model]

[I-D.ietf-idr-bgp-prefix-sid]

[I-D.ietf-idr-segment-routing-te-policy]

[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]

[I-D.ietf-rtgwg-routing-types]

[I-D.ietf-spring-segment-routing]
[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,

the Network Configuration Protocol (NETCONF)", RFC 6020,
DOI 10.17487/RFC6020, October 2010,

and A. Bierman, Ed., "Network Configuration Protocol
(NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/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,

8.2. Informative References

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG",
RFC 7951, DOI 10.17487/RFC7951, August 2016,

Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017,

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.decrane@orange.com

Zhichun Jiang  
Tencent  
Email: zcjiang@tencent.com
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 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 July 18, 2019.

Copyright Notice

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

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

Table of Contents

1.  Introduction ..................................................  3
2.  BGP Extra Extended Community Attribute ......................  3
3.  Transitivity ....................................................  4
4.  Capability ........................................................  5
5.  Constrained Route Distribution ...............................  5
6.  IPv6-Address-Specific Extra Extended Community type ......  7
7.  IPv4-Address-Specific Extra Extended Community type ......  7
8.  AS-Specific Extra Extended Community type ..................  8
9.  Route Target Extra Extended Community Sub-Type ............  9
10. EVPN Extra Extended Community type .......................... 10
11. EVPN Route Target Extra Extended Community sub-types ..... 10
12. EVPN ES-Import Route Target Extra Extended Community sub-type 11
13. EVPN ESI-EVI Route Target Extra Extended Community sub-type . 11
14. EVPN Overlay Route Target Extra Extended Community sub-type . 12
15. Duplicate XXC ..................................................... 14
16. Error Handling .................................................. 14
17. Security Considerations ......................................... 15
18. IANA Considerations ............................................. 15
   18.1. Registry: BGP Extra Extended Community Types ............ 15
   18.2. Registry: IPv6-Address-Specific Extra Extended Community
         Sub-Types .................................................... 15
   18.3. Registry: IPv4-Address-Specific Extra Extended Community
         Sub-Types .................................................... 16
   18.4. Registry: AS-Specific Extra Extended Community Sub-Types 16
   18.5. Registry: EVPN Extra Extended Community Sub-Types .... 16
19. References ....................................................... 17
   19.1. Normative References ....................................... 17
   19.2. Informative References .................................... 18
Authors’ Addresses ................................................. 18
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 an 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 offered 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 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 contains 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:
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 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
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
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 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 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].
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.
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

<table>
<thead>
<tr>
<th>Range</th>
<th>Allocation Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-31</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>32-47</td>
<td>Experimental</td>
</tr>
<tr>
<td>48-63</td>
<td>Standards Action</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>IPv6-Address-Specific</td>
<td>This RFC</td>
</tr>
<tr>
<td>1</td>
<td>IPv4-Address-Specific</td>
<td>This RFC</td>
</tr>
<tr>
<td>2</td>
<td>AS-Specific</td>
<td>This RFC</td>
</tr>
<tr>
<td>6</td>
<td>EVPN</td>
<td>This RFC</td>
</tr>
</tbody>
</table>

18.2. Registry: IPv6-Address-Specific Extra Extended Community Sub-Types
<table>
<thead>
<tr>
<th>Range</th>
<th>Allocation Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-191</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>192-255</td>
<td>IETF Review</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Route Target</td>
<td>RFC4360</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Range</th>
<th>Allocation Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-191</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>192-255</td>
<td>IETF Review</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Route Target</td>
<td>RFC4360</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Range</th>
<th>Allocation Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-191</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>192-255</td>
<td>IETF Review</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Route Target</td>
<td>RFC4360</td>
</tr>
</tbody>
</table>

### 18.5. Registry: EVPN Extra Extended Community Sub-Types

<table>
<thead>
<tr>
<th>Range</th>
<th>Allocation Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-191</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>192-255</td>
<td>IETF Review</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EVPN AS-Specific Route Target</td>
<td>This RFC</td>
</tr>
<tr>
<td>2</td>
<td>EVPN IPv4-Specific Route Target</td>
<td>This RFC</td>
</tr>
<tr>
<td>3</td>
<td>EVPN IPv6-Specific Route Target</td>
<td>This RFC</td>
</tr>
<tr>
<td>4</td>
<td>EVPN ES-Import Route Target</td>
<td>This RFC</td>
</tr>
<tr>
<td>5</td>
<td>EVPN ESI-EVI Route Target</td>
<td>This RFC</td>
</tr>
<tr>
<td>6</td>
<td>EVPN Overlay Route Target</td>
<td>This RFC</td>
</tr>
</tbody>
</table>
19. References

19.1. Normative References


19.2. Informative References

[I-D.ietf-bess-evpn-igmp-mld-proxy]

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
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

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 April 6, 2020.

Copyright Notice

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

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

Table of Contents

1. Introduction ........................................... 3
   1.1. Goals and approach ................................. 3
   1.2. Note to RFC Editor ................................... 4
   1.3. Terminology ......................................... 5
   1.4. Abbreviations ........................................ 5
2. Model overview ........................................... 5
   2.1. BGP protocol configuration ........................... 6
   2.2. Policy configuration overview .......................... 9
   2.3. BGP RIB overview ...................................... 10
      2.3.1. Local Routing .................................. 12
      2.3.2. Pre updates per-neighbor ......................... 12
      2.3.3. Post updates per-neighbor ....................... 12
      2.3.4. Pre route advertisements per-neighbor .......... 12
      2.3.5. Post route advertisements per-neighbor .......... 12
3. Relation to other YANG data models ......................... 12
4. Security Considerations .................................. 13
5. IANA Considerations ...................................... 13
6. YANG modules ............................................ 14
   6.1. URI Registration ..................................... 14
   6.2. YANG Module Name Registration ........................ 14
7. Structure of the YANG modules ............................. 15
   7.1. Main module and submodules for base items ........... 15
   7.2. BGP types ........................................... 64
   7.3. BGP policy data ....................................... 75
   7.4. RIB modules .......................................... 86
8. Examples ................................................ 120
   8.1. Creating BGP Instance ................................. 120
   8.2. Neighbor Address Family Configuration ............... 121
   8.3. IPv6 Neighbor Configuration ........................... 123
   8.4. VRF Configuration .................................... 124
9. Contributors ............................................. 125
10. Acknowledgements ....................................... 126
11. References .............................................. 126
   11.1. Normative references ................................ 126
   11.2. Informative references ............................... 128
Appendix A. How to add a new AFI and Augment a Module .......... 129
Appendix B. How to deviate a module .......................... 131
Authors’ Addresses ......................................... 132
1. Introduction

This document describes a YANG [RFC7950] data model for the BGP-4 [RFC4271] protocol, including various protocol extensions, policy configuration, as well as defining key operational state data, including 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 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], and BGP Prefix Origin Validation [RFC6811].

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 [I-D.ietf-rtgwg-policy-model]. The model conforms to the NMDA [RFC8342] architecture and has support for configuring Bidirectional Forward Detection (BFD) [RFC5880] for fast next hop liveliness check.

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

o 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.

2019-10-03 with the actual date of the publication of this document.

RFC ZZZZ, where ZZZZ is the number assigned to A YANG Data Model for Routing Policy Management [I-D.ietf-rtgwg-policy-model].

RFC AAAA, where AAAA is the number assigned to BGP Monitoring Protocol [RFC7854].

RFC BBBB, where BBBB is the number assigned to YANG Data Model for Bidirectional Forward Detection [I-D.ietf-bfd-yang].
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

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFI</td>
<td>Address Family Identifier</td>
</tr>
<tr>
<td>BFD</td>
<td>Bidirectional Forward Detection</td>
</tr>
<tr>
<td>NLRI</td>
<td>Network Layer Reachability Information</td>
</tr>
<tr>
<td>NMDA</td>
<td>Network Management Datastore Architecture</td>
</tr>
<tr>
<td>RIB</td>
<td>Routing Information Base</td>
</tr>
<tr>
<td>SAFI</td>
<td>Subsequent Address Family Identifier</td>
</tr>
<tr>
<td>VRF</td>
<td>Virtual Routing and Forwarding</td>
</tr>
</tbody>
</table>

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.
- multiprotocol configuration -- configuration affecting individual address-families within BGP Multiprotocol Extensions for BGP-4 [RFC4760].
- neighbor configuration -- configuration affecting an individual neighbor within BGP.
- neighbor multiprotocol configuration -- configuration affecting individual address-families for a neighbor within BGP.
o policy configuration -- hooks for application of the policies defined in A YANG Data Model for Routing Policy Management [I-D.ietf-rtgwg-policy-model] that act on routes sent (received) to (from) peers or other routing protocols and BGP-specific policy features.

o 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 {graceful-restart}?  
      +--rw use-multiple-paths
      +--rw route-selection-options
      +--rw afi-safis
      +--rw apply-policy
      +--ro total-paths? uint32
      +--ro total-prefixes? uint32
    +--rw neighbors
      +--rw neighbor* [remote-address]
        +--n established
        +--n backward-transition
        +--rw clear-neighbors {clear-neighbors}?
    +--rw peer-groups
      +--rw peer-group* [peer-group-name]
    +--rw interfaces
      +--rw interface* [name]
    +--ro rib
      +--ro attr-sets
      +--ro communities
      +--ro ext-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 to 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-neighbors 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 following address-families are currently supported by the model:
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 [I-D.ietf-rtgwg-policy-model], 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 specific neighbor or group.

module: ietf-bgp-policy
  augment /rpol:routing-policy/rpol:defined-sets:
    +--rw bgp-defined-sets
      ...
  augment /rpol:routing-policy/rpol:policy-definitions
    /rpol:policy-definition/rpol:statements/rpol:statement
      /rpol:conditions:
        +--rw bgp-conditions
          ...
  augment /rpol:routing-policy/rpol:policy-definitions
    /rpol:policy-definition/rpol:statements/rpol:statement
      /rpol: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.

A 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* [afi-safi-name]
        +--ro afi-safi-name identityref
    +--ro ipv4-unicast
      +--ro loc-rib
        +--ro routes
          +--ro route* [prefix origin path-id]
            ...  
          +--ro clear-routes {clear-routes}?  
            ...  
        +--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 clear-routes {clear-routes}?  
            ...  
        +--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 loc-rib table may contain multiple routes for a given prefix, with an attribute to indicate which was selected as the best path. Note that multiple paths may be used or advertised even if only one path is marked as best, e.g., when using BGP add-paths. An implementation may choose to mark multiple paths in the RIB as best path by setting the flag to true for multiple entries.

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.

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].
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. These are the subtrees and data nodes and their sensitivity/vulnerability:

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

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:

BGP OPSEC [RFC7454] describes several policies that can be used to secure a BGP. In particular, it recommends securing the underlying TCP session and to use 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 can be augmented with TCP-Authentication Option (TCP-AO) [RFC5925], or other methods to secure a BGP session, and will be developed in a future version of this draft.

5. IANA Considerations

This document registers three URIs and three YANG modules.
5.1. URI Registration

in the IETF XML registry [RFC3688] [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made:


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

5.2. YANG Module Name Registration

This document registers three YANG module in the YANG Module Names registry YANG [RFC6020].

name: ietf-bgp
prefix: bgp
reference: RFC XXXX

name: ietf-bgp-policy
prefix: bp
reference: RFC XXXX

name: ietf-bgp-types
prefix: bt
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-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 multiprotocol BGP
7. Structure of the YANG modules

The YANG model can be subdivided between the main module for base items, types, policy data, and the RIB module.

7.1. Main module and submodules for base items

<CODE BEGINS> file "ietf-bgp@2019-10-03.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 rpol;
  reference
    "RFC ZZZZ, A YANG Data Model for Routing Policy Management";
}
import ietf-interfaces {
  prefix if;
  reference
    "RFC 8343, A YANG Data Model for Interface Management.";
}
import ietf-bgp-types {
  prefix bt;
  reference
    "RFC XXXX, BGP YANG Model for Service Provider Network.";
}
import ietf-bfd-types {
  prefix bfd;
  reference
    "RFC BBBB, YANG Data Model for Bidirectional Forward Detection.";
}
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-common;
include ietf-bgp-common-multiprotocol;
include ietf-bgp-common-structure;
include ietf-bgp-neighbor;
include ietf-bgp-peer-group;
include ietf-bgp-rib-types;
include ietf-bgp-rib;
include ietf-bgp-rib-ext;
include ietf-bgp-rib-attributes;
include ietf-bgp-rib-table-attributes;
include ietf-bgp-rib-tables;

organization
  "IETF IDR Working Group";
contact
  "WG Web: <http://tools.ietf.org/wg/idr>"
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 ]
```

```
revision 2019-10-03 {
  description
    "Initial Version";
  reference
    "RFC XXXX, BGP Model for Service Provider Network ";
}
```

```
/*
 * Identity
 */

identity bgp {
  base rt:routing-protocol;
  description
    "BGP protocol.";
}
```
/* Feature(s) */

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

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

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

/* 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 distance (or preference) assigned to routes received from different sources (external, internal, and local).";
    leaf external {
        type uint8 {
            range "1..255";
        }
        description "Administrative distance for routes learned from external BGP (eBGP).";
    }
    leaf internal {
        type uint8 {
            range "1..255";
        }
        description "Administrative distance 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 graceful-restart;
  description
    "Parameters relating the 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 "afi-safi-name";
    description
      "AFI,SAFI configuration available for the
      neighbour or group";
    uses mp-afi-safi-config;
    uses state;
    container graceful-restart {
      if-feature graceful-restart;
      description
        "Parameters relating to BGP graceful-restart";
      uses mp-afi-safi-graceful-restart-config;
    }
  }
}
uses route-selection-options;
uses global-group-use-multiple-paths;
uses mp-all-afi-safi-list-contents;
}
}
uses rpol: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 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 {
        range "0..65535";
      }
      config false;
      description
      "The local port for the TCP connection between the BGP peers.";
    }

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

    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 open confirm or the established state."
reference
  "RFC 4271, Section 4.2, ‘BGP Identifier’.";
}
leaf remote-address {
  type inet:ip-address;
description
  "The remote IP address of this entry’s BGP peer."
}
leaf remote-port {
  type inet:port-number {
    range "0..65535";
  }
  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 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, Section 8.1.2.";

}

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.";
}

uses neighbor-group-config;
uses route-selection-options;

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, Section 8.1.2.";

leaf last-established {
    type uint64;
    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).";
}

leaf-list supported-capabilities {
    type identityref {
        base bt:bgp-capability;
    }
    config false;
    description "BGP capabilities negotiated as supported with the peer";
}

leaf negotiated-hold-time {
    type decimal64 {
        fraction-digits 2;
    }
    config false;
    description "The negotiated hold-time for the BGP session";
}
leaf last-error {
    type binary {
        length "2";
    }
    // notification does not like non-config statement.
    // config false;
    description
        "The last error code and subcode seen by this
         peer on this connection. If no error has
         occurred, this field is zero. Otherwise, the
         first byte of this two byte OCTET STRING
         contains the error code, and the second byte
         contains the subcode.";
    reference
        "RFC 4271, Section 4.5.";
}
leaf fsm-established-time {
    type yang:gauge32;
    units "seconds";
    config false;
    description
        "This timer indicates how long (in
         seconds) this peer has been in the
         established state or how long
         since this peer was last in the
         established state. It is set to zero when
         a new peer is configured or when the router is
         booted.";
    reference
        "RFC 4271, Section 8.";
}
container timers {
    description
        "Timers related to a BGP neighbor";
    uses neighbor-group-timers-config;
}
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.";
}

leaf erroneous-update-messages {
    type uint32;
    config false;
    description
        "The number of BGP UPDATE messages for which the
        treat-as-withdraw mechanism has been applied based on
        erroneous message contents";
}

container graceful-restart {
    if-feature graceful-restart;
    description
        "Parameters relating the 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 unset after all NLRI have been advertised to the peer, and the End-of-RIB (EOR) marker has been unset";
}

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";
}

uses structure-neighbor-group-logging-options;
uses structure-neighbor-group-ebgp-multihop;
uses structure-neighbor-group-route-reflector;
uses structure-neighbor-group-as-path-options;
uses structure-neighbor-group-add-paths;
uses bgp-neighbor-use-multiple-paths;
uses rpol:apply-policy-group;
container afi-safis {
    description
    "Per-address-family configuration parameters associated
    with the neighbor";
    uses bgp-neighbor-afi-safi-list;
}

container statistics {
    leaf established-transitions {
        type yang:counter64;
        config false;
        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";
    }

    leaf fsm-established-transitions {
        type yang:counter32;
        config false;
        description
        "The total number of times the BGP FSM
        transitioned into the established state
        for this peer.";
        reference
        "RFC 4271, Section 8.";
    }

    container messages {
        config false;
        description
        "Counters for BGP messages sent and received from the
        neighbor";

        leaf in-total-messages {
            type yang:counter32;
            config false;
            description
            "The total number of messages received
            from the remote peer on this connection.";
            reference
            "RFC 4271, Section 4.";
        }
    }
}
leaf out-total-messages {
  type yang:counter32;
  config false;
  description
      "The total number of messages transmitted to
      the remote peer on this connection.";
  reference
      "RFC 4271, Section 4.";
}

leaf in-update-elapsed-time {
  type yang:gauge32;
  units "seconds";
  config false;
  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, Section 4.3. 
      RFC 4271, Section 8.2.2, Established state.";
}

container sent {
  description
      "Counters relating to BGP messages sent to the
      neighbor";
  uses bgp-neighbor-counters-message-types-state;
}

container received {
  description
      "Counters for BGP messages received from the
      neighbor";
  uses bgp-neighbor-counters-message-types-state;
}

container queues {
  config false;
  description
      "Counters related to queued messages associated with
      the BGP neighbor";
  leaf input {
    type uint32;
    description
      "The total number of messages transmitted to
      the remote peer on this connection.";
  reference
      "RFC 4271, Section 4.";
  }
"The number of messages received from the peer currently queued";
}

leaf output {
  type uint32;
  description
    "The number of messages queued to be sent to the peer";
}

container clear-statistics {
  if-feature "clear-statistics";

  action clear {
    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.";
      }
    }

    description
      "Clear statistics action command.";
  }

  description
    "Statistics per neighbor.";
}

notification established {
  description
    "The established event is generated when the BGP FSM enters the established state.";

  leaf remote-address {
    type leafref {

leaf last-error {
  type leafref {
    path "../../neighbor/last-error";
  }
  description
  "The last error code and subcode seen by this peer on this connection. If no error has occurred, this field is zero. Otherwise, the first byte of this two byte OCTET STRING contains the error code, and the second byte contains the subcode.";
  reference
  "RFC 4271, Section 4.5.";
}

leaf session-state {
  type leafref {
    path "../../neighbor/session-state";
  }
  description
  "The BGP peer connection state.";
  reference
  "RFC 4271, Section 8.2.2.";
}

notification backward-transition {
  description
  "The backward-transition event is generated when the BGP FSM moves from a higher numbered state to a lower numbered state.";
  leaf remote-addr {
    type leafref {
      path "../../neighbor/remote-address";
    }
    description
    "IP address of the neighbor that went away from established state.";
  }
}
leaf last-error {
  type leafref {
    path "../../neighbor/last-error";
  }
  description
  "The last error code and subcode seen by this peer on this connection. If no error has occurred, this field is zero. Otherwise, the first byte of this two byte OCTET STRING contains the error code, and the second byte contains the subcode."
  reference
  "RFC 4271, Section 4.5."
}

leaf session-state {
  type leafref {
    path "../../neighbor/session-state";
  }
  description
  "The BGP peer connection state."
  reference
  "RFC 4271, Section 8.2.2."
}

container clear-neighbors {
  if-feature "clear-neighbors";

  action clear {
    input {
      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."
      }
    }
    description
  }
}
"Clear neighbors action.";
}

container peer-groups {
  description
    "Configuration for BGP peer-groups";
  uses bgp-peer-group-list;
}

container interfaces {
  list interface {
    key "name";

    leaf name {
      type if:interface-ref;
      description
        "Reference to the interface within the routing instance.";
    }

    container bfd {
      if-feature "bt:bfd";

      uses bfd:client-cfgParms;
      description
        "BFD client configuration.";
      reference
        "RFC BBBB - YANG Data Model for Bidirectional Forwarding Detection.";
    }

    description
      "List of interfaces within the routing instance.";
  }

  description
    "Interface specific parameters.";
}

  uses rib;
}

<CODE ENDS>
prefix "bgp";
}

import ietf-bgp-types {
    prefix bt;
    reference
        "RFC XXXX: BGP Model for Service Provider Network.";
}
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.";
}

organization
    "IETF IDR Working Group";

contact
    "WG Web:  <http://tools.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 pfrc.org).";

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.";

revision "2019-10-03" {
    description
        "Initial Version";
    reference
        "RFC XXXX, BGP Model for Service Provider Network.";
}

/*
 * Features.
*/
feature damping {

description
  "Weighted route dampening is supported.";
}

grouping neighbor-group-timers-config {
  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, Section 8.2.2. This is the value used to initialize the ‘ConnectRetryTimer’.";
}

  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
  }
}
leaf keepalive {
  type uint16 {
    range "0..21845";
  }
  units "seconds";
  default "30";
  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. The value of this object will only
determine the KEEPALIVE messages’ frequency relative to
the value specified in configured value for hold-time.

If the value of this object is zero (0), no periodical
KEEPALIVE messages are sent to the peer after the BGP
connection has been established. The suggested value for
this timer is 30 seconds.;

The actual time interval for the KEEPALIVE messages is
indicated by operational value of keepalive. That value
of this object is calculated by this BGP speaker such that,
when compared with hold-time, it has the same proportion
that keepalive has, compared with hold-time. A
reasonable maximum value for this timer would be one third
of that of hold-time.";
  reference
    "RFC 4271, Section 4.4.
    RFC 4271, Section 10.";
}

leaf min-as-origination-interval {
  type uint16 {
    range "0..max";
  }
  units "seconds";
  default "15";
  description
    "Time interval (in seconds) for the MinASOriginationInterval
timer. The suggested value for this timer is 15 seconds.";
  reference
    "RFC 4271, Section 9.2.1.2.
    RFC 4271, 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, Section 9.2.1.1.
    RFC 4271, Section 10.";
}

grouping neighbor-group-config {
  description "Neighbor level configuration items.";
  leaf remote-as {
    type inet:as-number;
    description "The remote autonomous system number received in
      the BGP OPEN message.";
    reference
      "RFC 4271, Section 4.2.";
  }
  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 peer-type {
    type bt:peer-type;
    description
"Explicitly designate the peer or peer group as internal (iBGP) or external (eBGP).";

leaf remove-private-as {
    // could also make this a container with a flag to enable
    // remove-private and separate option. here, option implies
    // remove-private is enabled.
    type bt:remove-private-as-option;
    description
    "Remove private AS numbers from updates sent to peers - when
    this leaf is not specified, the AS_PATH attribute should be
    sent to the peer unchanged";
}

container route-flap-damping {
    if-feature 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 yang:gauge32;
  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 yang:gauge32;
  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 yang:gauge32;
  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.";
}
"Routes learned via BGP are subject to weighted route dampening."

leaf send-community {
  if-feature "bt:send-communities";
  type bt:community-type;
  description
    "When supported, this tells the router to propagate any prefixes that are attached to this community. The value of 0 implies 'none'.";
}

leaf description {
  type string;
  description
    "An optional textual description (intended primarily for use with a peer or group";
}

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

  leaf tcp-mss {
    type uint16;
    description
      "Sets the max segment size for BGP TCP sessions.";
  }

  leaf mtu-discovery {
    type boolean;
    default false;
    description
      "Turns path mtu discovery for BGP TCP sessions on (true) or off (false)";
  }

  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 local-address {
    type union {
        type inet:ip-address;
        type leafref {
            path "../../interfaces/interface/name";
        }
    }
    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."
}

// TODO: Better form of authentication of the BGP session should
// be added here. It can be in the form of TCP-AO [RFC 5925],
// IPsec, or any other protocol deemed desirable.
leaf auth-password {
    type string;
    description "Configures an MD5 authentication password for use with
    neighboring devices."
}
}

grouping graceful-restart-config {
    description "Configuration parameters relating to BGP graceful restart."

    leaf enabled {
        type boolean;
        description "Enable or disable the graceful-restart capability."
    }

    leaf restart-time {
        type uint16 {
            range 0..4096;
        }
        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 held 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.";
    }
  }
}

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 to 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 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";
}
"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.";
}

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

  leaf total-prefixes {
    type uint32;
    config false;
    description
    "Total number of BGP prefixes received within the context";
  }
}

<CODE ENDS>
import ietf-routing-policy {
    prefix rpol;
}

include ietf-bgp-common;

// meta
organization
    "IETF IDR Working Group";
contact
    "WG Web:  <http://tools.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)"

description
    "This sub-module contains groupings that are related to support
     for multiple protocols in BGP. The groupings are common across
     multiple contexts.";
revision "2019-10-03" {
    description
        "Initial Version";
    reference
        "RFC XXX, BGP Model for Service Provider Network.";
}

grouping mp-afi-safi-graceful-restart-config {
    description
        "BGP graceful restart parameters that apply on a per-AFI-SAFI
         basis";
    leaf enabled {
        type boolean;
        default false;
        description
            "This leaf indicates whether graceful-restart is enabled for
             this AFI-SAFI";
    }
}

grouping mp-afi-safi-config {
    description
        "Configuration parameters used for all BGP AFI-SAFIs";
leaf afi-safi-name {
  type identityref {
    base "bt:afi-safi-type";
  }
  description "AFI,SAFI";
}

leaf enabled {
  type boolean;
  default false;
  description "This leaf indicates whether the IPv4 Unicast AFI,SAFI is enabled for the neighbour 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 rpol:apply-policy-group;
  container ipv4-unicast {
    when ".../afi-safi-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 ".../afi-safi-name = 'bt:ipv6-unicast'" {
      description "Include this container for IPv6 Unicast specific configuration";
    }
    description
"IPv6 unicast configuration options";

// include common IPv[4] unicast options
uses mp-ipv4-ipv6-unicast-common;

// placeholder for IPv6 unicast specific configuration
}

contAINER ipv4-labeled-unicast {
  when ".../afi-safi-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
}

contAINER ipv6-labeled-unicast {
  when ".../afi-safi-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
}

cONTAINER l3vpn-ipv4-unicast {
  when ".../afi-safi-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 "../afi-safi-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 "../afi-safi-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 "../afi-safi-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 "../afi-safi-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 "../afi-safi-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-safis {
  description


"Grouping for configuration common to all AFI,SAFI";

container prefix-limit {
  description "Parameters relating to the prefix limit for the AFI-SAFI";
  leaf max-prefixes {
    type uint32;
    description "Maximum number of prefixes that will be accepted from the neighbour";
  }
  leaf shutdown-threshold-pct {
    type bt:percentage;
    description "Threshold on number of prefixes that can be received from a neighbour before generation of warning messages or log entries. Expressed as a percentage of max-prefixes";
  }
  leaf restart-timer {
    type uint32;
    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 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 neighbour(s)";
  }
}

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>

<CODE BEGINS> file "ietf-bgp-common-structure@2019-10-03.yang"
submodule ietf-bgp-common-structure {
  yang-version "1.1";
  belongs-to ietf-bgp {
    prefix "bgp";
  }

  import ietf-bgp-types { prefix bt; }

import ietf-routing-policy { prefix rpol; }
include ietf-bgp-common-multiprotocol;
include ietf-bgp-common;

// meta
organization
"IETF IDR Working Group";

contact
"WG Web: <http://tools.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 pfrc.org).";

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

revision "2019-10-03" {
  description
  "Initial Version";
  reference
  "RFC XXX, BGP Model for Service Provider Network.";
}

grouping structure-neighbor-group-logging-options {
  description
  "Structural grouping used to include error handling
  configuration and state for both BGP neighbors and groups";

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 out of ESTABLISHED state is desirable,";"
but documenting all backward transitions is problematic, and should be avoided.


grouping structure-neighbor-group-ebgp-multihop {
    description
        "Structural grouping used to include eBGP multi-hop configuration and state for both BGP neighbors and peer groups";

container ebgp-multihop {
    description
        "eBGP multi-hop parameters for the BGP 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";
    }
}

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 group";
    reference
        "RFC 4456: BGP Route Reflection.";

    leaf route-reflector-cluster-id {

Jethanandani, et al. Expires April 6, 2020
when ".//route-reflector-client = 'false'";
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 no-client-reflect {
  type boolean;
  default "false";
  description
  "When set to 'true', this disables route redistribution
  by the Route Reflector. It is set 'true' when the client is
  fully meshed to prevent sending of redundant route
  advertisements.";
  reference
  "TODO: Add reference when IETF writes a draft describing
  this.";
}

leaf route-reflector-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."
}

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"
}

grouping structure-neighbor-group-add-paths {
    description
    "Structural grouping used to include ADD-PATHs configuration
     and state for both BGP neighbors and peer groups"
    container add-paths {
        description
        "Parameters relating to the advertisement and receipt of
         multiple paths for a single NLRI (add-paths)"
        reference
        "RFC 7911: ADD-PATH."
        leaf receive {
            type boolean;
            default false;
            description
            "Enable ability to receive multiple path advertisements for
             an NLRI from the neighbor or group"
        }
        choice send {
            default "all";
            description
            "Choice of sending the max. number of paths or to send all.";
            case max {
                leaf max {
                    type uint8;
                    description
                    "Number of paths to send."
                }
                leaf min {
                    type uint8;
                    description
                    "Minimum number of paths to send."
                }
            }
        }
    }
}
"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 "/rpol:routing-policy/rpol:policy-definitions/" + 
    "rpol:policy-definition/rpol:name";
  }
  description
  "A reference to a routing policy which can be used to restrict the prefixes for which add-paths is enabled";
}
}

<CODE ENDS>
"IETF IDR Working Group";

contact
  "WG Web:   <http://tools.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 pfrc.org).

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

revision "2019-10-03" {
  description
    "Initial Version";
  reference
    "RFC XXX, BGP Model for Service Provider Network.";
}

grouping bgp-peer-group-afi-safi-list {
  description
    "List of address-families associated with the BGP peer-group";

  list afi-safi {
    key "afi-safi-name";

    description
      "AFI, SAFI configuration available for the
       neighbour or group";

    uses mp-afi-safi-config;

    container graceful-restart {
      if-feature graceful-restart;
      description
        "Parameters relating to BGP graceful-restart";

      uses mp-afi-safi-graceful-restart-config;
    }

    uses route-selection-options;
    uses global-group-use-multiple-paths;
    uses mp-all-afi-safi-list-contents;
  }
}

grouping bgp-peer-group-base {
  description
    "Parameters related to a BGP group.";

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

  uses neighbor-group-config;

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

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

  container graceful-restart {
    if-feature graceful-restart;
    description
      "Parameters relating the graceful restart mechanism for BGP.";
    uses graceful-restart-config;
  }

  uses structure-neighbor-group-ebgp-multihop;
  uses structure-neighbor-group-route-reflector;
  uses structure-neighbor-group-as-path-options;
  uses structure-neighbor-group-add-paths;
  uses global-group-use-multiple-paths;
  uses rpol:apply-policy-group;

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

grouping bgp-peer-group-list {
    description
        "The list of BGP peer groups";

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

        uses bgp-peer-group-base;
    }
}

<CODE BEGINS> file "ietf-bgp-neighbor@2019-10-03.yang"
submodule ietf-bgp-neighbor {
    yang-version "1.1";
    belongs-to ietf-bgp {
        prefix "bgp";
    }

    // Include the common submodule
    include ietf-bgp-common;
    include ietf-bgp-common-multiprotocol;
    include ietf-bgp-peer-group;
    include ietf-bgp-common-structure;

    // meta
    organization
        "IETF IDR Working Group";

    contact
        "WG Web:  <http://tools.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 pfrc.org).";

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

    revision "2019-10-03" {

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. This value is overridden by any more specific configuration value.";
        }

        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 bgp-neighbor-counters-message-types-state {
    description
    "Grouping of BGP message types, included for re-use across counters";

    leaf updates-received {
        type uint64;
    }

leaf updates-sent {
  type uint64;
  description
    "Number of BGP UPDATE messages sent to this neighbor";
  reference
    "RFC 4273 - bgpPeerOutUpdates";
}

leaf messages-received {
  type uint64;
  description
    "Number of BGP messages received from this neighbor";
  reference
    "RFC 4273 - bgpPeerInTotalMessages";
}

leaf messages-sent {
  type uint64;
  description
    "Number of BGP messages received from this neighbor";
  reference
    "RFC 4273 - bgpPeerOutTotalMessages";
}

leaf notification {
  type uint64;
  description
    "Number of BGP NOTIFICATION messages indicating an error condition has occurred exchanged.";
}

grouping bgp-neighbor-afi-safi-list {
  description
    "List of address-families associated with the BGP neighbor";

  list afi-safi {
    key "afi-safi-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 BGP session";

leaf received {
    type uint32;
    description
        "The number of prefixes received from the neighbor";
}

leaf sent {
    type uint32;
    description
        "The number of prefixes advertised to the neighbor";
}

leaf installed {
    type uint32;
    description
        "The number of advertised prefixes installed in the
        Loc-RIB";
}

container graceful-restart {
    if-feature bgp:graceful-restart;
    description
        "Parameters relating to BGP graceful-restart";

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";
}

leaf advertised {
   type boolean;
   config false;
   description
   "This leaf indicates whether the ability to support
   graceful-restart has been advertised to the peer";
}
}

uses mp-all-afi-safi-list-contents;
uses bgp-neighbor-use-multiple-paths;
}
}

<CODE ENDS>

7.2. BGP types

<CODE BEGINS> file "ietf-bgp-types@2019-10-03.yang"
module ietf-bgp-types {
   yang-version "1.1";
   namespace "urn:ietf:params:xml:ns:yang:ietf-bgp-types";
   prefix "bt";
   import ietf-inet-types {
      prefix inet;
   }
   // meta
   organization "IETF IDR Working Group";
   contact
   "WG Web:  <http://tools.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 pfrc.org).";
description
  "This module contains general data definitions for use in BGP
policy. It can be imported by modules that make use of BGP
attributes";

revision "2019-10-03" {
  description
    "Initial Version";
  reference
    "RFC XXX, BGP Model for Service Provider Network.";
}

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";
}

identity route-refresh {
  base bgp-capability;
  description
    "The BGP route-refresh functionality";
  reference
    "RFC2918";
}

identity asn32 {
  base bgp-capability;
  description
    "4-byte (32-bit) AS number functionality";
  reference
    "RFC6793";
}

identity graceful-restart {
  base bgp-capability;
  description
    "Graceful restart functionality";
  reference
    "RFC4724";
}
identity add-paths {
    base bgp-capability;
    description
        "BGP add-paths";
    reference
        "RFC 7911.";
}

identity afi-safi-type {
    description
        "Base identity type for AFI,SAFI tuples for BGP-4";
    reference
        "RFC4760 - multi-protocol extensions for BGP-4";
}

identity ipv4-unicast {
    base afi-safi-type;
    description
        "IPv4 unicast (AFI,SAFI = 1,1)";
    reference
        "RFC4760";
}

identity ipv6-unicast {
    base afi-safi-type;
    description
        "IPv6 unicast (AFI,SAFI = 2,1)";
    reference
        "RFC4760";
}

identity ipv4-labeled-unicast {
    base afi-safi-type;
    description
        "Labeled IPv4 unicast (AFI,SAFI = 1,4)";
    reference
        "RFC3107";
}

identity ipv6-labeled-unicast {
    base afi-safi-type;
    description
        "Labeled IPv6 unicast (AFI,SAFI = 2,4)";
    reference
        "RFC3107";
}

identity l3vpn-ipv4-unicast {

base afi-safi-type;
description
  "Unicast IPv4 MPLS L3VPN (AFI,SAFI = 1,128)";
reference
"RFC4364";
}

identity l3vpn-ipv6-unicast {
  base afi-safi-type;
description
  "Unicast IPv6 MPLS L3VPN (AFI,SAFI = 2,128)";
reference
"RFC4659";
}

identity l3vpn-ipv4-multicast {
  base afi-safi-type;
description
  "Multicast IPv4 MPLS L3VPN (AFI,SAFI = 1,129)";
reference
"RFC6514";
}

identity l3vpn-ipv6-multicast {
  base afi-safi-type;
description
  "Multicast IPv6 MPLS L3VPN (AFI,SAFI = 2,129)";
reference
"RFC6514";
}

identity l2vpn-vpls {
  base afi-safi-type;
description
  "BGP-signalled VPLS (AFI,SAFI = 25,65)";
reference
"RFC4761";
}

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

identity bgp-well-known-std-community {
  description
  "Base identity for reserved communities within the standard

community space defined by RFC1997. 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 bi-lateral peer autonomous systems. An AS may also filter received NLRI from bilateral peer sessions when they are tagged with this community value";
    reference "RFC 3765: NOPEER Community for BGP.";
}
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."
}

/∗
 * Features.
 */

feature send-communities {

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

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"
}

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

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";
}
typedef bgp-std-community-type {
// TODO: further refine restrictions and allowed patterns
// 4-octet value:
// <as number> 2 octets
// <community value> 2 octets
type union {
  type uint32 {
    // per RFC 1997, 0x00000000 - 0x0000FFFF and 0xFFFF0000 -
    // 0xFFFFFFFF are reserved
    range "65536..4294901759"; // 0x00010000..0xFFFFEFFFF
  }
  type string {
    pattern "([0-9]+):([0-9]+)"
  }
}
description
"Type definition for standard community attributes";
reference
"RFC 1997 - BGP Communities Attribute";
}

typedef bgp-ext-community-type {
// TODO: needs more work to make this more precise given the
// variability of extended community attribute specifications
// 8-octet value:
// <type> 2 octects
// <value> 6 octets
type union {
  type string {
    // Type 1: 2-octet global and 4-octet local
    // (AS number) (Integer)
    pattern 
      '\(6[0-5][0-5][0-3][0-5][1-5][0-9][0-9]\)\|\(1-[9][0-9]\)[1,4][0-9]\)\|\(0-9\)\)
      +
      '\(4[0-2][0-9][0-4][0-9][0-6][0-7][0-2][0-9][0-9][0-6]\)\|\(1-[3][0-9]\)[0-9]\)\|\(1-[9]\)([0-9]\)[1,7]\)\|\(0-9\)\)\|\(1-9\)
      +
    }
  }
  type string {
    // Type 2: 4-octet global and 2-octet local
    // (ipv4-address) (integer)
    pattern 
      '\(0-9\)\|\(1-[9]\)[0-9]\)\|\(1-9\)[0-9]\)\|\(2[0-4][0-9]\)\|\(25[0-5]\)\)
      +
      '\(6[0-5][0-5][0-3][0-5][1-5][0-9]\)\|\(0-9\)\)\|\(4\)
      +
    }
  }
  type string {
    // route-target with Type 1
  }
}
// route-target: (ASN): (local-part)

pattern 'route\-target:(6[0-5][0-5][0-3][0-5]|' +
  '[1-5][0-9][0-9][1,4][0-9]):' +
  '[40-2][0-9][0-9][0-6][0-7][0-2][0-9][0-6]|' +
  '[1-3][0-9][0-9][1-9]([0-9][1,7])?[0-9][1-9]):';

// route-target: (IPv4): (local-part)

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[0-9][0-9]|2[0-4][0-9]|' +
  '25[0-5]):' +
  '6[0-9][0-5][0-3][0-5]|1-5[0-9][0-9]{4}|' +
  '[1-9][0-9][0-9][1,4]|0-9))';

// route-origin: (ASN): (local-part)

pattern 'route\-origin:(6[0-5][0-5][0-3][0-5]|' +
  '[1-5][0-9][0-9][1,4][0-9]):' +
  '[40-2][0-9][0-9][0-6][0-7][0-2][0-9][0-6]|' +
  '[1-3][0-9][0-9][1-9]([0-9][1,7])?[0-9][1-9]):';

// route-origin: (IPv4): (local-part)

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[0-9][0-9]|2[0-4][0-9]|' +
  '25[0-5]):' +
  '6[0-9][0-5][0-3][0-5]|1-5[0-9][0-9]{4}|' +
  '[1-9][0-9][0-9][1,4]|0-9))';

typedef bgp-community-regexp-type {
  // TODO: needs more work to decide what format these regexps can
  // take.
  type string;
  description
    "Type definition for communities specified as regular
    expression patterns";
}

typedef bgp-community-regexp-type {
  // TODO: needs more work to decide what format these regexps can
  // take.
  type string;
  description
    "Type definition for extended community attributes";
  reference
    "RFC 4360 - BGP Extended Communities Attribute";
}

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";
}

typedef peer-type {
  type enumeration {
    enum internal {
      description "internal (iBGP) peer";
    }  
    enum external {
      description "external (eBGP) peer";
    }  
    enum confederation {
      description "Confederation as peer";
    }  
  }
  description "Labels a peer or peer group as explicitly internal, external or confederation.";
}

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."
}

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"
}

typedef percentage {
    type uint8 {
        range "0..100";
    }
    description
    "Integer indicating a percentage value"
}

typedef rr-cluster-id-type {
    type union {
        type uint32;
        type inet:ipv4-address;
    }
    description
    "Union type for route reflector cluster ids:
    option 1: 4-byte number
    option 2: IP address"
}

typedef community-type {
    type bits {
        bit standard {
            position 0;
            description
            """
"Send only standard communities."
reference
"RFC 1997: BGP Communities Attribute."
}
bit extended {
  description
  "Send only extended communities."
  reference
  "RFC 4360: BGP Extended Communities Attribute."
}
bit large {
  description
  "Send only large communities."
  reference
  "RFC 8092: BGP Large Communities Attribute."
}

description
"Type describing variations of community attributes. The community types can be combined and a value of 0 implies 'none'."
}
</CODE ENDS>

7.3. BGP policy data

<CODE BEGINS> file "ietf-bgp-policy@2019-10-03.yang"
module ietf-bgp-policy {
  yang-version "1.1";
  prefix "bp";

  // import some basic types
  import ietf-inet-types {
    prefix inet;
  }
  import ietf-routing-policy {
    prefix rpol;
  }
  import ietf-bgp-types {
    prefix bt;
  }
  import ietf-routing-types {
    prefix rt-types;
  }

organization
  "IETF IDR Working Group";

contact
  "WG Web: <http://tools.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 pfrc.org).";

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.";

revision "2019-10-03" {
  description
    "Initial Version";
  reference
    "RFC XXX, BGP Model for Service Provider Network.";
}

// 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 uint32;
        type string {
            pattern "^[+-]\([0-9]{1,8}|[0-3][0-9]{1,9}|4[0-1][0-9]{1,8}|428[0-9]{1,7}|429[0-3][0-9]{1,6}|42948[0-9]{1,5}|42949[0-5]{1,4}|429496[0-6]{1,3}|429497[12]{1,2}|42949728[0-9]{1,5}|42949729[0-5]\)$";
        }
    }

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-IGP 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 apply.

// augment statements

augment "/rpol:routing-policy/rpol: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 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;
            mandatory true;
            description "Name / label of the community set -- this is used to reference the set in match conditions";
        }

        leaf-list member {
            type union {
                type bt:bgp-std-community-type;
                type bt:bgp-community-regexp-type;
                type bt:bgp-well-known-community-type;
            }
            description
        }
    }

    }
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 rt-types:route-target;
        type bt:bgp-community-regexp-type;
      }
      description
      "Members of the extended community set";
    }
  }
}

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 {
  // TODO: need to refine typedef for AS path expressions
  type string;
  description "AS path expression -- list of ASes in the set";
}

grouping set-community-action-common {
  description "Common leaves for set-community and set-ext-community actions";
  leaf method {
    type enumeration {
      enum inline {
        description "The extended communities are specified inline as a list";
      }
      enum reference {
        description "The extended communities are specified by referencing a defined ext-community set";
      }
    }
    description "Indicates the method used to specify the extended communities for the set-ext-community action";
  }
  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.";
  }
}

  description "BGP policy conditions added to routing policy module";
container bgp-conditions {
    description
    "Top-level container for BGP specific policy conditions ";

    leaf med-eq {
        type uint32;
        description
        "Condition to check if the received MED value is equal to
         the specified value";
    }

    leaf origin-eq {
        type bt:bgp-origin-attr-type;
        description
        "Condition to check if the route origin is equal to the
         specified value";
    }

    leaf-list next-hop-in {
        type inet:ip-address-no-zone;
        description
        "List of next hop addresses to check for in the route
         update";
    }

    leaf-list afi-safi-in {
        type identityref {
            base bt:afi-safi-type;
        }
        description
        "List of address families which the NLRI may be within";
    }

    leaf local-pref-eq {
        type uint32;
        // TODO: add support for other comparisons if needed
        description
        "Condition to check if the local pref attribute is equal to
         the specified value";
    }

    leaf route-type {
        // TODO: verify extent of vendor support for this comparison
        type enumeration {
            enum internal {
                description "route type is internal";
            }
            enum external {
                description "route type is 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";
}

container as-path-length {

description
"Value and comparison operations for conditions based on the length of the AS path in the route update";
}

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
"/rpol:routing-policy/rpol:defined-sets/" + "bp:bgp-defined-sets/bp:community-sets/" + "bp:community-set(bp:name);
}
description
"References a defined community set";
}

uses rpol: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
"/rpol:routing-policy/rpol:defined-sets/" +
uses rpol: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 "/rpol:routing-policy/rpol:defined-sets/" +
         "bp:bgp-defined-sets/bp:as-path-sets/" +
         "bp:as-path-set/bp:name";
    }
    description "References a defined AS path set";
    uses rpol:match-set-options-group;
  }
}

augment "/rpol:routing-policy/rpol:policy-definitions/" +
"rpol:policy-definition/rpol:statements/rpol:statement/" +
"rpol: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 update";
    }
}
leaf set-next-hop {
  type bgp-next-hop-type;
  description
    "Set the next-hop attribute in the route update";
}

leaf set-med {
  type bgp-set-med-type;
  description
    "Set the med metric attribute in the route update";
}

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 local AS number to the AS
      path. The value should be between 1 and the maximum
      supported by the implementation.";
  }
}

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).";
  uses set-community-action-common;
  container inline {
    when "../method = 'inline'" {
      description
        "Active only when the set-community method is inline";
    }
    description
      "Set the community values for the action inline with
      a list.";
    leaf-list communities {

type union {
  type bt:bgp-std-community-type;
  type bt:bgp-well-known-community-type;
} description "Set the community values for the update inline with a list.";
}
}

container reference {
  when "../method = 'reference'" {
    description "Active only when the set-community method is reference";
  }
  description "Provide a reference to a defined community set for the set-community action";
}

leaf community-set-ref {
  type leafref {
  }
  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).";
}

uses set-community-action-common;
container inline {
  when "../method = 'inline'" {
    description "Active only when the set-community method is inline";
  }
  description "Set the extended community values for the action inline with a list.";
}
leaf-list communities {
    type union {
        type rt-types:route-target;
        type bt:bgp-well-known-community-type;
    }
    description
    "Set the extended community values for the update inline
    with a list."
}
}

container reference {
    when "../method = 'reference'" {
        description
        "Active only when the set-community method is reference";
    }
    description
    "Provide a reference to an extended community set for the
    set-ext-community action"

    leaf ext-community-set-ref {
        type leafref {
            path
            "/rpol:routing-policy/rpol:defined-sets/
             "bp:bgp-defined-sets/bp:ext-community-sets/
             "bp:ext-community-set/bp:name";
        }
        description
        "References a defined extended community set by name"
    }
}

// rpc statements

// notification statements
}

7.4. RIB modules

<CODE BEGINS> file "ietf-bgp-rib@2019-10-03.yang"

submodule ietf-bgp-rib {
    yang-version "1.1";
    belongs-to ietf-bgp {
        prefix "br";
    }
}

<CODE ENDS>
import ietf-bgp-types {
  prefix "bt";
  reference
    "RFC XXXX: BGP YANG Model for Service Provider Networks.";
}
import ietf-inet-types {
  prefix inet;
  reference
    "RFC 6991: Common YANG Types.";
}
import ietf-yang-types {
  prefix yang;
  reference
    "RFC 6991: Common YANG Types.";
}
import ietf-routing-types {
  prefix "rt";
  reference
    "RFC 8294: Routing Area YANG Types.";
}
include ietf-bgp-rib-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
include ietf-bgp-rib-attributes;

// groupings of annotations for each route or table
include ietf-bgp-rib-table-attributes;

organization
  "IETF IDR Working Group";

contact
  "WG Web: <http://tools.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)

description
"Defines a submodule for representing BGP routing table (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 loc-rib table may contain multiple routes for a given prefix, with an attribute to indicate which was selected as the best path. Note that multiple paths may be used or advertised even if only one path is marked as best, e.g., when using BGP add-paths. An implementation may choose to mark multiple paths in the RIB as best path by setting the flag to true for multiple entries.

adj-rib-in-pre: This 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.

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."

revision "2019-10-03" {
  description
    "Initial Version";
  reference
    "RFC XXXX, BGP YANG Model for Service Provider Network."
}

grouping rib {
  description
    "Grouping for rib.";
}
container rib {
    config false;
}

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.";
    }
    leaf origin {
        type bt:bgp-origin-attr-type;
        description
            "BGP attribute defining the origin of the path
            information.";
    }
    leaf atomic-aggregate {
        type boolean;
        description
            "BGP attribute indicating that the prefix is an atomic
            aggregate; i.e., the peer selected a less specific
            route without selecting a more specific route that is
            included in it.";
        reference
            "RFC 4271: Section 5.1.6.";
    }
    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";
        reference
            "RFC 4271: Section 5.1.6.";
    }
}
leaf med {
    type uint32;
    description "BGP multi-exit discriminator attribute used in BGP route selection process";
    reference "RFC 4271: 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: Section 5.1.5.";
}

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";
}
container aggregator {
  config false;
  description "BGP attribute indicating the prefix has been aggregated by the specified AS and router."
  reference "RFC 4271: Section 5.1.7.";

  leaf as {
    type inet:as-number;
    description "AS number of the autonomous system that performed the aggregation.";
  }

  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";
  }

  leaf address {
    type inet:ipv4-address;
    description "IP address of the router that performed the aggregation.";
  }
}

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 route 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.

 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 {
  key "type";
  config false;
  uses bgp-as-path-attr;
  description "List of AS PATH segments";
}
```

```
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 {
    key "type";
    config false;
    uses bgp-as-path-attr;
    description "List of AS PATH segments";
  }
}
```

```
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."
  }

  leaf-list ext-community {
    type rt:route-target;
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."
  }
}
"RFC 4360 - BGP Extended Communities Attribute";
}
}

container afi-safis {
  config false;
  description
  "Enclosing container for address family list";

  list afi-safi {
    key "afi-safi-name";
    description
      "List of afi-safi types.";

    leaf afi-safi-name {
      type identityref {
        base bt:afi-safi-type;
      }
      description "AFI,SAFI name.";
    }

    container ipv4-unicast {
      when ".../afi-safi-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";

      uses ipv4-loc-rib;
      uses ipv4-adj-rib;
    }

    container ipv6-unicast {
      when ".../afi-safi-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";

      uses ipv6-loc-rib;
      uses ipv6-adj-rib;
    }
  }
}
grouping rib-ext-route-annotations {
    description "Extended annotations for routes in the routing tables";

    leaf reject-reason {

    }
type union {
  type identityref {
    base bgp-not-selected-bestpath;
  }
  type identityref {
    base bgp-not-selected-policy;
  }
}
description
  "Indicates the reason the route is not used, either due to policy filtering or bestpath selection";
}

<CODE BEGINS> file "ietf-bgp-rib-types@2019-10-03.yang"
submodule ietf-bgp-rib-types {
  yang-version "1.1";
  belongs-to ietf-bgp {
    prefix "br";
  }

  organization
    "IETF IDR Working Group";

  contact
    "WG Web:  <http://tools.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 pfrc.org).";

  description
    "Defines identity and type definitions associated with the BGP RIB modules";

  revision "2019-10-03" {
    description
      "Initial Version";
    reference
      "RFC XXXX, BGP Model for Service Provider Network.";
  }

  identity invalid-route-reason {

description
 "Base identity for reason code for routes that are rejected as invalid. Some derived entities are based on BMP v3";
reference
 "RFC 7854: BGP Monitoring Protocol.";
)

identity invalid-cluster-loop {
 base invalid-route-reason;
 description
 "Route was invalid due to CLUSTER_LIST loop";
}

identity invalid-as-loop {
 base invalid-route-reason;
 description
 "Route was invalid due to AS_PATH loop";
}

identity invalid-originator {
 base invalid-route-reason;
 description
 "Route was invalid due to ORIGINATOR_ID, e.g., update has local router as originator";
}

identity invalid-confed {
 base invalid-route-reason;
 description
 "Route was invalid 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 - 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 - 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 - 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 - 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 - Section 9.1.2.2 (c)";
}

identity prefer-external {
    base bgp-not-selected-bestpath;
    description "Route source is via IGP, rather than EGP.";
    reference "RFC 4271 - 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 - 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, or router id";
    reference
"RFC 4271 - 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 - 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 apply import policies";
}
}

<CODE ENDS>

<CODE BEGINS> file "ietf-bgp-rib-attributes@2019-10-03.yang"
submodule ietf-bgp-rib-attributes {
    yang-version "1.1";
    belongs-to ietf-bgp {
        prefix "br";
    }

    // import some basic types
    import ietf-bgp-types {
        prefix bgpt;
    }

    import ietf-inet-types {
        prefix inet;
    }

    include ietf-bgp-rib-types;

    // meta
    organization
        "IETF IDR Working Group";
This submodule contains common data definitions for BGP attributes for use in BGP RIB tables.

revision "2019-10-03" {
  description "Initial version";
  reference "RFC XXXX: BGP YANG Model for Service Provider Network";
}

grouping bgp-as-path-attr {
  description "Data for representing BGP AS-PATH attribute";

  leaf type {
    type identityref {
      base bgpt: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-community-attr-state {
  description "Common definition of BGP community attributes";

  leaf-list community {
    type union {
      type bgpt:bgp-well-known-community-type;
      type bgpt:bgp-std-community-type;
    }
  }
}
description
  "List of standard or well-known BGP community attributes.";
}
}

grouping bgp-unknown-attr-flags-state {
  description
  "Operational state data for path attribute flags";

  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)";
}

} grouping bgp-unknown-attr-state {
  description
    "Operational state data for path attributes not shared across route entries, common to LOC-RIB and Adj-RIB";

  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-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.";

      uses bgp-unknown-attr-flags-state;
      uses bgp-unknown-attr-state;
    }
  }
}

grouping bgp-loc-rib-attr-state {
  description
  "Path attributes that are not expected to be shared across
  route entries, specific to LOC-RIB";
}

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";
  }
}
<CODE BEGINS> file "ietf-bgp-rib-table-attributes@2019-10-03.yang"
submodule ietf-bgp-rib-table-attributes {
  yang-version "1.1";
  belongs-to ietf-bgp {
    prefix "br";
  }

  // import some basic types
  import ietf-yang-types {
    prefix types;
    reference "RFC 6991, Common YANG Data Types.";
  }

  include ietf-bgp-rib-types;

  organization "IETF IDR Working Group";

  contact "WG Web:  <http://tools.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";

  description "This submodule contains common data definitions for data
               related to a RIB entry, or RIB table.";

  revision "2019-10-03" {
    description "Initial version.";
    reference "RFC XXXX: BGP YANG Model for Service Provider Network.";
  }

  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 types: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 valid-route {
  type boolean;
  description
  "Indicates that the route is considered valid by the local router";
}

leaf invalid-reason {
  type identityref {
    base invalid-route-reason;
  }
  description
  "If the route is rejected as invalid, this indicates the reason.";
}

} grouping bgp-loc-rib-route-annotations-state {
  description
  "Data definitions for information attached to routes in the LOC-RIB";

  // placeholder for route metadata specific to the LOC-RIB
}

} 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;
    description
    "Current path was selected as the best path.";
  }

}
grouping bgp-common-table-attrs-state {
    description
        "Common attributes attached to all routing tables";

    // placeholder for metadata associated with all tables
}

grouping bgp-common-table-attrs-top {
    // no enclosing container as this data will fit under an
    // existing LOC-RIB container

    uses bgp-common-table-attrs-state;
    description
        "Operational state data for data related to the entire
         LOC-RIB";
}

<CODE ENDS>

<CODE BEGINS> file "ietf-bgp-rib-tables@2019-10-03.yang"
submodule ietf-bgp-rib-tables {
    yang-version "1.1";
    belongs-to ietf-bgp {
        prefix "br";
    }

    // 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";
}

include ietf-bgp-rib-ext;
include ietf-bgp-rib-attributes;
include ietf-bgp-rib-table-attributes;

organization
  "IETF IDR Working Group";

contact
  "WG Web:  <http://tools.ietf.org/wg/idr>
  WG List:  <idr@ietf.org>
  Editor:  Mahesh Jethanandani (mjethanandani@gmail.com)
  Authors: Keyur Patel,
           Mahesh Jethanandani,
           Susan Hares";

description
  "This submodule contains structural data definitions for
  BGP routing tables.";

revision "2019-10-03" {
  description
    "Initial Version";
  reference
    "RFC XXXX, BGP YANG Model for Service Provider Network.";
}

/*
* Feature(s)
*/

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

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";
}

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";
    }
  }
}
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;
    // TODO: YANG does not allow default values for key
    // default 0;
    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.";
  }
}

grouping clear-routes {
  description "Action to clear BGP routes.";

  container clear-routes {
    if-feature "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-loc-rib {
    description
    "Top-level grouping for IPv4 routing tables";
    container loc-rib {
        config false;
        description
        "Container for the IPv4 BGP LOC-RIB data";
        uses bgp-common-table-attrs-top;
        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-loc-rib-attr-state;
uses bgp-common-route-annotations-state;
uses bgp-loc-rib-route-annotations-state;
uses bgp-unknown-attr-top;
uses rib-ext-route-annotations;
}

uses clear-routes;
}
}

grouping ipv6-loc-rib {
  description
    "Top-level grouping for IPv6 routing tables";
  container loc-rib {
    config false;
    description
      "Container for the IPv6 BGP LOC-RIB data";
    uses bgp-common-table-attrs-top;
  }
  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-loc-rib-attr-state;
uses bgp-common-route-annotations-state;
uses bgp-loc-rib-route-annotations-state;
uses bgp-unknown-attr-top;
uses rib-ext-route-annotations;

uses clear-routes;
}
}
}

grouping ipv4-adj-rib-common {
  description
    "Common structural grouping for each IPv4 adj-RIB table";

  uses bgp-common-table-attrs-top;

  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;
}

uses clear-routes;
}
}

grouping ipv4-adj-rib-in-post {
    description
        "Common structural grouping for the IPv4 adj-rib-in post-policy table";
    uses bgp-common-table-attrs-top;
    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 ipv4-adj-rib {
  description
  "Top-level grouping for Adj-RIB table";

  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 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 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 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 ipv4-adj-rib-common;
}
}

}

}

grouping ipv6-adj-rib-common {
  description
  "Common structural grouping for each IPv6 adj-RIB table";

  uses bgp-common-table-attrs-state;

  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;
    }
  }
}

uses clear-routes;
}

grouping ipv6-adj-rib-in-post {
  description
  "Common structural grouping for the IPv6 adj-rib-in
  post-policy table";
  uses bgp-common-table-attrs-state;
  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;
    }
  }
}

grouping ipv6-adj-rib {
  description
  "Top-level grouping for Adj-RIB table";
  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 ipv6-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 ipv6-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 ipv6-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 ipv6-adj-rib-common;
}
<CODE BEGINS> file "ietf-bgp-rib-table-attributes@2019-10-03.yang"
submodule ietf-bgp-rib-table-attributes { 
yang-version "1.1";
belongs-to ietf-bgp { 
  prefix "br";
}

// import some basic types
import ietf-yang-types { 
  prefix types;
  reference
  "RFC 6991, Common YANG Data Types.";
}

include ietf-bgp-rib-types;

organization
  "IETF IDR Working Group";

contact
  "WG Web: <http://tools.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)";

description
  "This submodule contains common data definitions for data
  related to a RIB entry, or RIB table."

revision "2019-10-03" { 
  description
    "Initial version.";
  reference
    "RFC XXXX: BGP YANG Model for Service Provider Network.";
}

grouping bgp-common-route-annotations-state {

leaf last-modified {
  type types: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 valid-route {
  type boolean;
  description
  "Indicates that the route is considered valid by the local router";
}

leaf invalid-reason {
  type identityref {
    base invalid-route-reason;
  }
  description
  "If the route is rejected as invalid, this indicates the reason.";
}

// placeholder for route metadata specific to the LOC-RIB

// Data definitions for information attached to routes in the Adj-RIB-in post-policy table

leaf best-path {
  type boolean;
  description
  "Current path was selected as the best path.";
}
8. Examples

This section tries to show some examples in how the model can be used.

8.1. Creating BGP Instance

This example shows how to enable BGP with the IPv4 unicast address family, while adding one network to advertise.
8.2. Neighbor Address Family Configuration

This example shows how to configure a BGP peer, where the remote address is 192.0.2.1, the remote AS number is 64497, and the address family of the peer is IPv4 unicast.

[NOTE: '\' Line Wrapping for Formatting Only]

<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <control-plane-protocols>
      <control-plane-protocol>
        <name>BGP</name>
        <bgp xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
          <global>
            <as>64496</as>
            <afi-safis>
              <afi-safi>
              </afi-safi>
            </afi-safis>
          </global>
        </bgp>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>

This example shows a neighbor configuration with damping.

<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <control-plane-protocols>
      <control-plane-protocol>
        <name>BGP</name>
        <bgp xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
          <global>
            <as>64496</as>
            <afi-safis>
              <afi-safi>
              </afi-safi>
            </afi-safis>
          </global>
        </bgp>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>
<control-plane-protocols>
  <control-plane-protocol>
    <name>name:BGP</name>
    <bgp xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
      <global>
        <as>64496</as>
        <afi-safis>
          <afi-safi>
          </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>
          <description>"Peer Router B"</description>
          <afi-safis>
            <afi-safi>
            </afi-safi>
          </afi-safis>
        </neighbor>
      </neighbors>
    </bgp>
  </control-plane-protocol>
</control-plane-protocols>
8.3. IPv6 Neighbor Configuration

This example shows how to configure a BGP peer, where the remote peer has an IPv6 address, and uses non-default timers for hold-time and keepalive.

[note: ‘\’ line wrapping for formatting only]

```xml
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <control-plane-protocols>
      <control-plane-protocol>
        <name>BGP</name>
        <bgp xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
          <global>
            <as>64496</as>
            <afi-safis>
              <afi-safi>
              </afi-safi>
            </afi-safis>
          </global>
          <neighbors>
            <neighbor>
              <remote-address>2001:db8::</remote-address>
              <enabled>true</enabled>
              <peer-as>64497</peer-as>
              <description>"Peer Router B"</description>
              <timers>
                <hold-time>120</hold-time>
                <keepalive>70</keepalive>
              </timers>
              <afi-safis>
                <afi-safi>
                </afi-safi>
              </afi-safis>
            </neighbor>
          </neighbors>
        </bgp>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>
```
8.4. 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.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <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
              <name>BGP</name>
              <bgp
                xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
                <global>
                  <as>64496</as>
                  <afi-safis>
                    <afi-safi>
                      <afi-safi-name
                    </afi-safi>
                  </afi-safis>
                  </global>
                </bgp>
              </control-plane-protocol>
            </control-plane-protocols>
          </routing>
        </network-instance>
      </vrf-root>
    </network-instance>
  </network-instances>
</config>
```
</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
          <name>BGP</name>
          <bgp
            xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
            <global>
              <as>64496</as>
              <afi-safis>
                <afi-safi>
                  <afi-safi-name
                    xmlns:bt="urn:ietf:params:xml:ns:yang:ietf-bgp-types">
                    bt:ipv4-unicast
                  </afi-safi-name>
                </afi-safi>
                </afi-safis>
            </global>
          </bgp>
        </control-plane-protocol>
      </control-plane-protocols>
    </routing>
  </vrf-root>
</network-instance>
</network-instances>
</config>

9. Contributors

Previous versions of this document saw contributions from Anees Shaikh, Rob Shakir, Kevin D'Souza, Alexander Clemm, Aleksandr Zhadkin, and Xyfeng Liu.
10. 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, Jeff Haas, 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.

11. References

11.1. Normative references


11.2. Informative references

[I-D.ietf-bfd-yang]

[I-D.ietf-rtgwg-policy-model]

[RFC5082]

[RFC5880]
Appendix A. 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-routing {
    prefix rt;
    reference
      "RFC 8349, A YANG Data Model for Routing Management (NMDA Version)";
  }

  import ietf-bgp {
    prefix "bgp";
    reference
      "RFC XXXX: BGP YANG module for Service Provider Network.";
  }

  import ietf-bgp-types {
    prefix "bt";
  }
}
organization
"Newafi model group.";

contact
"abc@newafi.com";

description
"This YANG module defines and uses new AFI.";

revision 2019-10-03 {

description
"Creating new AFI and using in this model";

reference
"RFC XXXX: BGP YANG Model for Service Provider Network.";
}

identity foo {

description
"New AFI type foo.";
}

augment "/rt:routing/rt:control-plane-protocols/" +
"rt:control-plane-protocol/bgp:bgp/global/" +
"bgp:afi-safis/bgp:afi-safi" {

when "derived-from-or-self(bgp:afi-safi-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.";
}
}

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:afi-safi-name, 'foo')" {

description
"This augmentation is valid for a AFI/SAFI instance of 'foo';";
}

container foo {

description
Appendix B. 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 2019-10-03 {
        description
            "Creating NewCo deviations to ietf-bgp model";

        reference
            "RFC XXXX: BGP YANG module for Service Provider Network.";
    }

    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;
    }
}
Authors’ Addresses

Mahesh Jethanandani
VMware
Email: mjethanandani@gmail.com

Keyur Patel
Arrcus
CA
USA
Email: keyur@arrcus.com

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI  48176
USA
Email: shares@ndzh.com

Jeffrey Haas
Juniper Networks
Email: jhaas@pfrc.org
Applying BGP flowspec rules on a specific interface set
draft-ietf-idr-flowspec-interfaceset-04

Abstract

The BGP Flow Specification (flowspec) Network Layer Reachability Information (BGP NLRI) extension ([RFC5575]) 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 permit 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", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.
Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on December 30, 2018.

Copyright Notice

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

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

Table of Contents

1. Use case ................................................. 3
2. Interface specific filtering using BGP flowspec ........ 3
3. Interface-set extended community .......................... 4
4. Scaling of per-interface rules ............................ 6
5. Deployment Considerations .................................. 6
   5.1. Add-Paths ........................................... 6
   5.2. Inter-domain Considerations .......................... 6
6. Security Considerations .................................... 7
7. Acknowledgements .......................................... 7
8. IANA Considerations ....................................... 7
   8.1. FlowSpec Transitive Extended Communities ............. 7
   8.2. FlowSpec Non-Transitive Extended Communities ......... 7
   8.3. FlowSpec interface-set Extended Community ............ 8
   8.4. Allocation Advice to IANA ............................ 8
9. Normative References ....................................... 8

Authors’ Addresses .......................................... 9
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:

```
+-----------------------------------------------+
| 0x07 or 0x47 |      0x02     |    Autonomous System Number   : 
+-----------------------------------------------+
|     AS Number (cont.)         |O|I|      Group Identifier     | 
+-----------------------------------------------+
```

The flags are:

- **O**: if set, the flow specification rule MUST be applied in outbound direction to the interface set referenced by the following group-identifier.

- **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 [RFC5575] 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 [RFC5575] 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


Authors’ Addresses

Stephane Litkowski
Orange
Email: stephane.litkowski@orange.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
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

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

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

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

This Internet-Draft will expire on October 20, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents
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 /
  /
_______| ISP1 (AS1) |----------->| ISP2 (AS2)---------->
| prefix |           | prefix(P)           | route-leak(P)
( P) /  \\   /   \   \   \   \
------  prefix(P)  \   \\   \   \   \
   update  /
      /   \   \   \   \   \
     /     \   \   \   \   \
    /       \   \   \   \   \
   /         \   \   \   \   \
  /          \   \   \   \   \
 /            \   \   \   \   \
+-------------------+  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.
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 $L = \text{local ANS}$. 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.)
Internet-Draft     Route Leak Detection and Mitigation        April 2019

[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


6.2. Informative References


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
Abstract

This document defines a new BGP SAFI with a new NLRI in order to advertise a candidate path of a Segment Routing Policy (SR Policy). An SR Policy is a set of candidate paths, each consisting of one or more segment lists. The headend of an SR Policy may learn multiple candidate paths for an SR Policy. Candidate paths may be learned via a number of different mechanisms, e.g., CLI, NetConf, PCEP, or BGP. This document specifies the way in which BGP may be used to distribute candidate paths. New sub-TLVs for the Tunnel Encapsulation Attribute are defined.

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 6, 2020.
1. Introduction

Segment Routing (SR) allows a headend node to steer a packet flow along any path. Intermediate per-flow states are eliminated thanks to source routing [I-D.ietf-spring-segment-routing].

The headend node is said to steer a flow into a Segment Routing Policy (SR Policy).

The header of a packet steered in an SR Policy is augmented with the ordered list of segments associated with that SR Policy.

[I-D.ietf-spring-segment-routing-policy] details the concepts of SR Policy and steering into an SR Policy. These apply equally to the MPLS and SRv6 instantiations of segment routing.

[I-D.filsfils-spring-sr-policy-considerations] describes some of the implementation aspects of the SR Policy Headend Architecture and introduces the notion of an SR Policy Module (SRPM) that performs the functionality as highlighted in section 2 of [I-D.ietf-spring-segment-routing-policy]:

- The SRPM may learn multiple candidate paths for an SR Policy via various mechanisms (CLI, NetConf, PCEP or BGP).
- The SRPM selects the best candidate path for the SR Policy.
- The SRPM binds a BSID to the selected candidate path of the SR Policy.
- The SRPM installs the selected candidate path and its BSID in the forwarding plane.

This document specifies the way to use BGP to distribute one or more of the candidate paths of an SR Policy to the headend of that policy. The document identifies the functionality that resides in the BGP process and for the functionality which is outside the scope of BGP.
and lies within SRPM on the headend node, it refers to such, as appropriate.

This document specifies a way of representing SR Policies and their candidate paths in BGP UPDATE messages. BGP can then be used to propagate the SR Policies and candidate paths. The usual BGP rules for BGP propagation and "bestpath 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 defines a new BGP address family (SAFI). In UPDATE messages of that address family, the NLRI identifies an SR Policy, and the attributes encode the segment lists and other details of that SR Policy.

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 this document), PCEP, NETCONF or local policy configuration.

Typically, a controller defines the set of policies and advertise them to policy head-end routers (typically ingress routers). The policy advertisement uses BGP extensions defined in this document. The policy advertisement is, in most but not all of the cases, tailored for a specific policy head-end. In this case the advertisement may sent on a BGP session to that head-end 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 head-end over a BGP session to that head-end, 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 head-end. In these situations, BGP Route Reflectors may be used to propagate the advertisements, or it may be necessary for the advertisement to propagate through a sequence of one or more ASes. 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 head-end 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 new Subsequent Address Family Identifier (SAFI) whose NLRI identifies an SR Policy.
- A new Tunnel Type identifier for SR Policy, and a set of sub-TLVs to be inserted into the Tunnel Encapsulation Attribute (as defined in [I-D.ietf-idr-tunnel-encaps]) specifying segment lists of the SR Policy, 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 head-end of such SR Policy advertisement.
- The Color Extended Community (as defined in [I-D.ietf-idf-r tunnel-encaps]) and used in order to steer traffic into an SR Policy, as described in section 8.4 in [I-D.ietf-spring-segment-routing-policy]. This document (Section 3) modifies the format of the Color Extended Community by using the two leftmost bits of the RESERVED field.

1.1. Requirements Language

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

2. SR Policy Encoding

2.1. SR Policy SAFI and NLRI

A new SAFI is defined: the SR Policy SAFI, (codepoint 73 assigned by IANA (see Section 8) from the "Subsequent Address Family Identifiers (SAFI) Parameters" registry).

The SR Policy SAFI uses a new NLRI defined as follows:
<table>
<thead>
<tr>
<th>NLRI Length</th>
<th>1 octet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distinguisher</td>
<td>4 octets</td>
</tr>
<tr>
<td>Policy Color</td>
<td>4 octets</td>
</tr>
<tr>
<td>Endpoint</td>
<td>4 or 16 octets</td>
</tr>
</tbody>
</table>

where:

- **NLRI Length**: 1 octet of length expressed in bits as defined in [RFC4760].
- **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) multiple occurrences of the same SR Policy.
- **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 [I-D.ietf-spring-segment-routing-policy].
- **Endpoint**: 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 color and endpoint are used to automate the steering of BGP Payload prefixes on SR Policy as described in [I-D.ietf-spring-segment-routing-policy].

The NLRI containing the SR Policy is carried in a BGP UPDATE message [RFC4271] using BGP multiprotocol extensions [RFC4760] with an AFI of 1 or 2 (IPv4 or IPv6) and with a SAFI of 73 (assigned by IANA from the "Subsequent Address Family Identifiers (SAFI) Parameters" registry).

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; and 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 in [RFC4760].

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 does not modify the BGP propagation or bestpath selection rules.

It has to be noted that if several candidate paths of the same SR Policy (endpoint, color) are signaled via BGP to a head-end, it is recommended that each NLRI use 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. In addition, the originator information corresponding to the each candidate path, as described in section 2.4 in [I-D.ietf-spring-segment-routing-policy], is passed to the SRPM.

2.2. SR Policy and Tunnel Encapsulation Attribute

The content of the SR Policy is encoded in the Tunnel Encapsulation Attribute originally defined in [I-D.ietf-idr-tunnel-encaps] using a new Tunnel-Type TLV (codepoint is 15, assigned by IANA (see Section 8) from the "BGP Tunnel Encapsulation Attribute Tunnel Types" registry).

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
  ...

where:

  - SR Policy SAFI NLRI is defined in Section 2.1.
  - Tunnel Encapsulation Attribute is defined in [I-D.ietf-idr-tunnel-encaps].
  - Tunnel-Type is set to 15 (assigned by IANA from the "BGP Tunnel Encapsulation Attribute Tunnel Types" registry).
  - Preference, Binding SID, Priority, Policy Name, ENLP, Segment-List, Weight and Segment sub-TLVs are defined in this document.
  - Additional sub-TLVs may be defined in the future.

A Tunnel Encapsulation Attribute MUST NOT contain more than one TLV of type "SR Policy". If more than one TLV of type "SR Policy" appears, the update is considered malformed and the "treat-as-withdraw" strategy of [RFC7606] is applied.

Multiple occurrences of "Segment List" MAY be encoded within the same SR Policy.

Multiple occurrences of "Segment" MAY be encoded within the same Segment List.

2.3.  Remote Endpoint and Color

The Remote Endpoint and Color sub-TLVs, as defined in [I-D.ietf-idr-tunnel-encaps], MAY also be present in the SR Policy encodings.
The Remote Endpoint and Color Sub-TLVs 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 Remote Endpoint sub-TLV and the Color sub-TLV MUST be ignored by the BGP speaker.

2.4. SR Policy Sub-TLVs

This section defines the SR Policy sub-TLVs.

Preference, Binding SID, Segment-List, Priority, Policy Name and Explicit NULL Label Policy sub-TLVs are assigned from the "BGP Tunnel Encapsulation Attribute Sub-TLVs" registry.

Weight and Segment sub-TLVs are assigned from a new registry defined in this document and called: "SR Policy List Sub-TLVs". See Section 8 for the details of the registry.

2.4.1. Preference Sub-TLV

The Preference sub-TLV does not have any effect on the BGP bestpath selection or propagation procedures. The contents of this sub-TLV are used by the SRPM as described in section 2.7 in [I-D.ietf-spring-segment-routing-policy].

The Preference sub-TLV is optional and it MUST NOT appear more than once in the SR Policy. If the Preference sub-TLV appears more than once, the update is considered malformed and the "treat-as-withdraw" strategy of [RFC7606] is applied.

The Preference sub-TLV has following format:

```
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)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

where:

- **Type**: 12
- **Length**: 6.
- **Flags**: 1 octet of flags. None are defined at this stage. Flags SHOULD be set to zero on transmission and MUST be ignored on receipt.
2.4.2. Binding SID Sub-TLV

The Binding SID sub-TLV is not used by BGP. The contents of this sub-TLV are used by the SRPM as described in section 6 in [I-D.ietf-spring-segment-routing-policy].

The Binding SID sub-TLV is optional and it MUST NOT appear more than once in the SR Policy. If the Binding SID sub-TLV appears more than once, the update is considered malformed and the "treat-as-withdraw" strategy of [RFC7606] is applied.

The Binding SID sub-TLV has the following format:

```
  0                   1                   2                   3
  +---------------+---------------+---------------+---------------+
  |     Type      |   Length      |     Flags     |   RESERVED    |
  +---------------+---------------+---------------+---------------+
  |              Binding SID (variable, optional)                 |
  +---------------+---------------+---------------+---------------+
```

where:

- **Type**: 13
- **Length**: specifies the length of the value field not including Type and Length fields. Can be 2 or 6 or 18.
- **Flags**: 1 octet of flags. Following flags are defined (to be assigned by IANA from the registry "SR Policy Binding SID Flags" defined in this document Section 8.5):

```
  0 1 2 3 4 5 6 7
  +---------------+
  |S|I|           |
  +---------------+
```

where:

- **S-Flag**: This flag encodes the "Specified-BSID-only" behavior. It is used by SRPM as described in section 6.2.3 in [I-D.ietf-spring-segment-routing-policy].
* I-Flag: This flag encodes the "Drop Upon Invalid" behavior. It is used by SRPM as described in section 8.2 in [I-D.ietf-spring-segment-routing-policy].

* Unused bits in the Flag octet SHOULD be set to zero upon transmission and MUST be ignored upon receipt.

  o RESERVED: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt.

  o Binding SID: if length is 2, then no Binding SID is present.

  o If length is 6 then the Binding SID contains a 4-octet SID. Below format is used to encode the SID. TC, S, TTL (Total of 12 bits) are RESERVED and SHOULD 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
     +-----------------------------------------------+
     |          Label                        | TC  |S|       TTL     |
     +-----------------------------------------------+

If length is 18 then the Binding SID contains a 16-octet IPv6 SID.

2.4.3. Segment List Sub-TLV

The Segment List sub-TLV encodes a single explicit path towards the endpoint as described in section 5.1 in [I-D.ietf-spring-segment-routing-policy]. 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 length due to large number of segments. Therefore a 2-octet length is required. According to [I-D.ietf-idr-tunnel-encaps], the first bit of the sub-TLV codepoint defines the size of the length field. Therefore, for the Segment List sub-TLV a code point of 128 (or higher) is used. See Section 8 for details of codepoints allocation.

The Segment List sub-TLV is optional and MAY appear multiple times in the SR Policy. The ordering of Segment List sub-TLVs, each sub-TLV encoding a Segment List, does not matter.

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:
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.
- **RESERVED**: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt.
- **sub-TLVs**:
  - * 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 completely within the scope of SRPM as described in section 5 in [I-D.ietf-spring-segment-routing-policy].

### 2.4.3.1. Weight Sub-TLV

The Weight sub-TLV specifies the weight associated to a given segment list. The contents of this sub-TLV are used only by the SRPM as described in section 2.11 in [I-D.ietf-spring-segment-routing-policy].

The Weight sub-TLV is optional and it MUST NOT appear more than once inside the Segment List sub-TLV. If the Weight sub-TLV appears more than once, the update is considered malformed and the "treat-as-withdraw" strategy of [RFC7606] is applied.

The Weight sub-TLV has the following format:
where:

Type: 9 (to be assigned by IANA from the registry "SR Policy List Sub-TLVs" defined in this document).

Length: 6.

Flags: 1 octet of flags. None are defined at this stage. Flags SHOULD be set to zero on transmission and MUST be ignored on receipt.

RESERVED: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt.

2.4.3.2. Segment Sub-TLV

The Segment sub-TLV describes a single segment in a segment list (i.e., a single element of the explicit path). Multiple Segment sub-TLVs constitute an explicit path of the SR Policy.

The Segment sub-TLV is optional and MAY appear multiple times in the Segment List sub-TLV.

The Segment sub-TLV does not have any effect on the BGP bestpath selection or propagation procedures. The contents of this sub-TLV are used only by the SRPM as described in section 4 in [I-D.ietf-spring-segment-routing-policy].

[I-D.ietf-spring-segment-routing-policy] defines several types of Segments:
Type 1: SID only, in the form of MPLS Label
Type 2: SID only, in the form of IPv6 address
Type 3: IPv4 Node Address with optional SID
Type 4: IPv6 Node Address with optional SID for SR MPLS
Type 5: IPv4 Address + index with optional SID
Type 6: IPv4 Local and Remote addresses with optional SID
Type 7: IPv6 Address + index for local and remote pair with optional SID for SR MPLS
Type 8: IPv6 Local and Remote addresses with optional SID for SR MPLS
Type 9: IPv6 Node Address with optional SID for SRv6
Type 10: IPv6 Address + index for local and remote pair with optional SID for SRv6
Type 11: IPv6 Local and Remote addresses for SRv6

2.4.3.2.1. Type 1: SID only, in the form of MPLS Label

The Type-1 Segment Sub-TLV encodes a single SID in the form of an MPLS label. 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type     |   Length    |     Flags    |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Label                        | TC  |S|       TTL     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- **Type**: 1 (to be assigned by IANA from the registry "SR Policy List Sub-TLVs" defined in this document).
- **Length** is 6.
- **Flags**: 1 octet of flags as defined in Section 2.4.3.2.12.
- **RESERVED**: 1 octet of reserved bits. SHOULD be unset 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:
o The S bit SHOULD be zero upon transmission, and MUST be ignored upon reception.

o If the originator wants the receiver to choose the TC value, it sets the TC field to zero.

o If the originator wants the receiver to choose the TTL value, it sets the TTL field to 255.

o If the originator wants to recommend a value for these fields, it puts those values in the TC and/or TTL fields.

o 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.

2.4.3.2.2. Type 2: SID only, in the form of IPv6 address

The Type-2 Segment Sub-TLV encodes a single SRv6 SID in the form of an IPv6 address. The format is as follows:

```
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)                  //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

o Type: 2 (to be assigned by IANA from the registry "SR Policy List Sub-TLVs" defined in this document).

o Length is 18.

o Flags: 1 octet of flags as defined in Section 2.4.3.2.12.

o RESERVED: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt.

o SRv6 SID: 16 octets of IPv6 address.

The IPv6 Segment Identifier (SRv6 SID) is defined in [I-D.ietf-6man-segment-routing-header].
2.4.3.2.3. Type 3: IPv4 Node Address with optional SID

The Type-3 Segment Sub-TLV encodes an IPv4 node address, SR Algorithm and an optional SID in the form of an MPLS label. The format is as follows:

```
+----------------------------------+
|       Type       |   Length   |     Flags    |  SR Algorithm  |
|------------------+------------|-------------|--------------|
|                  |            |             |              |
+----------------------------------+
<table>
<thead>
<tr>
<th>IPv4 Node Address (4 octets)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SID (optional, 4 octets)</td>
</tr>
</tbody>
</table>
+----------------------------------+
```

where:

- **Type**: 3 (to be assigned by IANA from the registry "SR Policy List Sub-TLVs" defined in this document).
- **Length**: is 6 or 10.
- **Flags**: 1 octet of flags as defined in Section 2.4.3.2.12.
- **SR Algorithm**: 1 octet specifying SR Algorithm as described in section 3.1.1 in [I-D.ietf-spring-segment-routing], when A-Flag as defined in Section 2.4.3.2.12 is present. SR Algorithm is used by SRPM as described in section 4 in [I-D.ietf-spring-segment-routing-policy]. When A-Flag is not encoded, this field SHOULD be unset on transmission and MUST be ignored on receipt.
- **IPv4 Node Address**: a 4 octet IPv4 address representing a node.
- **SID**: 4 octet MPLS label.

The following applies to the Type-3 Segment sub-TLV:

- The IPv4 Node Address MUST be present.
- The SID is optional and specifies a 4 octet MPLS SID containing label, TC, S and TTL as defined in Section 2.4.3.2.1.
- If length is 6, then only the IPv4 Node Address is present.
- If length is 10, then the IPv4 Node Address and the MPLS SID are present.
2.4.3.2.4. Type 4: IPv6 Node Address with optional SID for SR MPLS

The Type-4 Segment Sub-TLV encodes an IPv6 node address, SR Algorithm and an optional SID in the form of an MPLS label. The format is as follows:

```
+---------------+---------------+---------------+---------------+
|     Type      |   Length      |     Flags     |  SR Algorithm |
+---------------+---------------+---------------+---------------+
//                        IPv6 Node Address (16 octets)  //</
+---------------+---------------+---------------+---------------+
|                SID (optional, 4 octets)                       |
+---------------+---------------+---------------+---------------+
```

where:

- **Type**: 4 (to be assigned by IANA from the registry "SR Policy List Sub-TLVs" defined in this document).
- **Length**: is 18 or 22.
- **Flags**: 1 octet of flags as defined in Section 2.4.3.2.12.
- **SR Algorithm**: 1 octet specifying SR Algorithm as described in section 3.1.1 in [I-D.ietf-spring-segment-routing], when A-Flag as defined in Section 2.4.3.2.12 is present. SR Algorithm is used by SRPM as described in section 4 in [I-D.ietf-spring-segment-routing-policy]. When A-Flag is not encoded, this field SHOULD be unset on transmission and MUST be ignored on receipt.
- **IPv6 Node Address**: a 16 octet IPv6 address representing a node.
- **SID**: 4 octet MPLS label.

The following applies to the Type-4 Segment sub-TLV:

- The IPv6 Node Address MUST be present.
- The SID is optional and specifies a 4 octet MPLS SID containing label, TC, S and TTL as defined in Section 2.4.3.2.1.
- If length is 18, then only the IPv6 Node Address is present.
- If length is 22, then the IPv6 Node Address and the MPLS SID are present.
2.4.3.2.5. Type 5: IPv4 Address + Local Interface ID with optional SID

The Type-5 Segment Sub-TLV encodes an IPv4 node address, a local interface Identifier (Local Interface ID) and an optional SID in the form of an MPLS label. The format is as follows:

```
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    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Local Interface ID (4 octets)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 IPv4 Node Address (4 octets)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               SID (optional, 4 octets)                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- Type: 5 (to be assigned by IANA from the registry "SR Policy List Sub-TLVs" defined in this document).
- Length is 10 or 14.
- Flags: 1 octet of flags as defined in Section 2.4.3.2.12.
- RESERVED: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt.
- Local Interface ID: 4 octets of interface index as defined in [I-D.ietf-pce-segment-routing].
- IPv4 Node Address: a 4 octet IPv4 address representing a node.
- SID: 4 octet MPLS label.

The following applies to the Type-5 Segment sub-TLV:

- The IPv4 Node Address MUST be present.
- The Local Interface ID MUST be present.
- The SID is optional and specifies a 4 octet MPLS SID containing label, TC, S and TTL as defined in Section 2.4.3.2.1.
- If length is 10, then the IPv4 Node Address and Local Interface ID are present.
2.4.3.2.6. Type 6: IPv4 Local and Remote addresses with optional SID

The Type-6 Segment Sub-TLV encodes an adjacency local address, an adjacency remote address and an optional SID in the form of an MPLS label. The format is as follows:

```
0                   1                   2                   3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Length      |     Flags     |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Local IPv4 Address (4 octets)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Remote IPv4 Address  (4 octets)                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     SID (optional, 4 octets)                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- **Type**: 6 (to be assigned by IANA from the registry "SR Policy List Sub-TLVs" defined in this document).
- **Length**: 10 or 14.
- **Flags**: 1 octet of flags as defined in Section 2.4.3.2.12.
- **RESERVED**: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt.
- **Local IPv4 Address**: a 4 octet IPv4 address.
- **Remote IPv4 Address**: a 4 octet IPv4 address.
- **SID**: 4 octet MPLS label.

The following applies to the Type-6 Segment sub-TLV:

- The **Local IPv4 Address** MUST be present and represents an adjacency local address.
- The **Remote IPv4 Address** MUST be present and represents the remote end of the adjacency.
o The SID is optional and specifies a 4 octet MPLS SID containing label, TC, S and TTL as defined in Section 2.4.3.2.1.

o If length is 10, then only the IPv4 Local and Remote addresses are present.

o If length is 14, then the IPv4 Local address, IPv4 Remote address and the MPLS SID are present.

2.4.3.2.7. Type 7: IPv6 Address + Interface ID for local and remote pair with optional SID for SR MPLS

The Type-7 Segment Sub-TLV encodes an IPv6 Link Local adjacency with IPv6 local node address, a local interface identifier (Local Interface ID), IPv6 remote node address, a remote interface identifier (Remote Interface ID) and an optional SID in the form of an MPLS label. The format is as follows:

```
  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    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Local Interface ID (4 octets)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                IPv6 Local Node Address (16 octets)          //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Remote Interface ID (4 octets)                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                IPv6 Remote Node Address (16 octets)         //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                SID (optional, 4 octets)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

o Type: 7 (to be assigned by IANA from the registry "SR Policy List Sub-TLVs" defined in this document).

o Length is 22, 26, 42 or 46.

o Flags: 1 octet of flags as defined in Section 2.4.3.2.12.

o RESERVED: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt.

o Local Interface ID: 4 octets of interface index as defined in [I-D.ietf-pce-segment-routing].
The following applies to the Type-7 Segment sub-TLV:

- The Local Interface ID and IPv6 Local Node Address MUST be present.
- The Remote Interface ID and Remote Node Address pair is optional. If Remote Interface ID is present, the Remote Node Address MUST be present as well. Similarly, if Remote Node Address is present, the Remote Interface ID MUST be present as well.
- The SID is optional and specifies a 4 octet MPLS SID containing label, TC, S and TTL as defined in Section 2.4.3.2.1.

- If length is 22, then the Local Interface ID and the Local IPv6 Address are present.
- If length is 26, then the Local Interface ID, Local IPv6 Address and the MPLS SID are present.
- If length is 42, then the Local Interface ID, Local IPv6 Node Address, Remote Interface ID, and the Remote IPv6 Node Address are present.
- If length is 46, then the Local Interface ID, Local IPv6 Node Address, Remote Interface ID, Remote IPv6 Node Address and the MPLS SID are present.

2.4.3.2.8. Type 8: IPv6 Local and Remote addresses with optional SID for SR MPLS

The Type-8 Segment Sub-TLV encodes an adjacency local address, an adjacency remote address and an optional SID in the form of an MPLS label. The format is as follows:
<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
<th>Flags</th>
<th>RESERVED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local IPv6 Address (16 octets)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remote IPv6 Address (16 octets)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SID (optional, 4 octets)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where:

- **Type**: 8 (to be assigned by IANA from the registry "SR Policy List Sub-TLVs" defined in this document).
- **Length**: 34 or 38.
- **Flags**: 1 octet of flags as defined in Section 2.4.3.2.12.
- **RESERVED**: 1 octet of reserved bits. **SHOULD** be unset on transmission and **MUST** be ignored on receipt.
- **Local IPv6 Address**: a 16 octet IPv6 address.
- **Remote IPv6 Address**: a 16 octet IPv6 address.
- **SID**: 4 octet MPLS label.

The following applies to the Type-8 Segment sub-TLV:

- **The Local IPv6 Address** **MUST** be present and represents an adjacency local address.
- **The Remote IPv6 Address** **MUST** be present and represents the remote end of the adjacency.
- **The SID** is optional and specifies a 4 octet MPLS SID containing label, TC, S and TTL as defined in Section 2.4.3.2.1.
- **If length is 34**, then only the IPv6 Local and Remote addresses are present.
- **If length is 38**, then IPv6 Local and Remote addresses and the MPLS SID are present.
2.4.3.2.9. Type 9: IPv6 Node Address with optional SRv6 SID

The Type-9 Segment Sub-TLV encodes an IPv6 node address, SR Algorithm and an optional SID in the form of an IPv6 address. The format is as follows:

<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
<th>Flags</th>
<th>SR Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv6 Node Address (16 octets)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

// SID (optional, 16 octets) //

where:

- **Type:** 10 (to be assigned by IANA from the registry "SR Policy List Sub-TLVs" defined in this document).
- **Length:** 18 or 34.
- **Flags:** 1 octet of flags as defined in Section 2.4.3.2.12.
- **SR Algorithm:** 1 octet specifying SR Algorithm as described in section 3.1.1 in \[I-D.ietf-spring-segment-routing\], when A-Flag as defined in Section 2.4.3.2.12 is present. SR Algorithm is used by SRPM as described in section 4 in \[I-D.ietf-spring-segment-routing-policy\]. When A-Flag is not encoded, this field SHOULD be unset on transmission and MUST be ignored on receipt.
- **IPv6 Node Address:** a 16 octet IPv6 address.
- **SID:** 16 octet IPv6 address.

The following applies to the Type-9 Segment sub-TLV:

- The IPv6 Node Address MUST be present.
- The SID is optional and specifies an SRv6 SID in the form of 16 octet IPv6 address.
- If length is 18, then only the IPv6 Node Address is present.
- If length is 34, then the IPv6 Node Address and the SRv6 SID are present.
2.4.3.2.10. Type 10: IPv6 Address + Interface ID for local and remote pair for SRv6 with optional SID

The Type-10 Segment Sub-TLV encodes an IPv6 Link Local adjacency with local node address, a local interface identifier (Local Interface ID), remote IPv6 node address , a remote interface identifier (Remote Interface ID) and an optional SID in the form of an IPv6 address. The format is as follows:

```
+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+
<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
<th>Flags</th>
<th>RESERVED</th>
</tr>
</thead>
</table>
+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+
|                 Local Interface ID (4 octets) |               |
+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+
//                IPv6 Local Node Address (16 octets) //
+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+
|                 Remote Interface ID (4 octets) |
+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+
//                IPv6 Remote Node Address (16 octets) //
+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+
//                SID (optional, 16 octets) //
+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+
```

where:

- Type: 11 (to be assigned by IANA from the registry "SR Policy List Sub-TLVs" defined in this document).
- Length is 22, 38, 42 or 58.
- Flags: 1 octet of flags as defined in Section 2.4.3.2.12.
- RESERVED: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt.
- Local Interface ID: 4 octets of interface index as defined in [I-D.ietf-pce-segment-routing].
- IPv6 Local Node Address: a 16 octet IPv6 address.
- Remote Interface ID: 4 octets of interface index as defined in [I-D.ietf-pce-segment-routing].
- IPv6 Remote Node Address: a 16 octet IPv6 address.
- SID: 16 octet IPv6 address.
The following applies to the Type-10 Segment sub-TLV:

- The Local Interface ID and the Local IPv6 Node Addresses MUST be present.
- The Remote Interface ID and Remote Node Address pair is optional. If Remote Interface ID is present, the Remote Node Address MUST be present as well. Similarly, if Remote Node Address is present, the Remote Interface ID MUST be present as well.
- The SID is optional and specifies an SRv6 SID in the form of 16 octet IPv6 address.
- If length is 22, then the Local Interface ID, Local IPv6 Node Address, are present.
- If length is 38, then the Local Interface ID, Local IPv6 Node Address and the SRv6 SID are present.
- If length is 42, then the Local Interface ID, Local IPv6 Node Address, Remote Interface ID, and the Remote IPv6 Node Address are present.
- If length is 58, then the Local Interface ID, Local IPv6 Node Address, Remote Interface ID, Remote IPv6 Node Address and the SRv6 SID are present.

2.4.3.2.11. Type 11: IPv6 Local and Remote addresses for SRv6 with optional SID

The Type-11 Segment Sub-TLV encodes an adjacency local address, an adjacency remote address and an optional SID in the form of IPv6 address. 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    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
//               Local IPv6 Address (16 octets)                //
//               Remote IPv6 Address (16 octets)              //
//                SID (optional, 16 octets)                    //
where:
```
- Type: 12 (to be assigned by IANA from the registry "SR Policy List Sub-TLVs" defined in this document).
- Length is 34 or 50.
- Flags: 1 octet of flags as defined in Section 2.4.3.2.12.
- RESERVED: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt.
- Local IPv6 Address: a 16 octet IPv6 address.
- Remote IPv6 Address: a 16 octet IPv6 address.
- SID: 16 octet IPv6 address.

The following applies to the Type-11 Segment sub-TLV:
- The Local IPv6 Node Address MUST be present.
- The Remote IPv6 Node Address MUST be present.
- The SID is optional and specifies an SRv6 SID in the form of 16 octet IPv6 address.
- If length is 34, then the Local IPv6 Node Address and the Remote IPv6 Node Address are present.
- If length is 50, then the Local IPv6 Node Address, the Remote IPv6 Node Address and the SRv6 SID are present.

2.4.3.2.12. Segment Flags

The Segment Types described above MAY contain following flags in the "Flags" field (codes to be assigned by IANA from the registry "SR Policy Segment Flags" defined in this document Section 8.6):

```
  0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-
|V|A|           |
+-+-+-+-+-+-+-+-+-
              ^
              |
              +-----+
```

where:

- V-Flag: This flag is used by SRPM for the purpose of "SID verification" as described in Section 5.1 in [I-D.ietf-spring-segment-routing-policy].
A-Flag: This flag indicates the presence of SR Algorithm id in the "SR Algorithm" field applicable to various Segment Types. SR Algorithm is used by SRPM as described in section 4 in [I-D.ietf-spring-segment-routing-policy].

Unused bits in the Flag octet SHOULD be set to zero upon transmission and MUST be ignored upon receipt.

The following applies to the Segment Flags:

- V-Flag is applicable to all Segment Types.
- A-Flag is applicable to Segment Types 3, 4 and 9. If A-Flag appears with any other Segment Type, it MUST be ignored.

2.4.4. Explicit NULL Label Policy Sub-TLV

In order to steer an unlabeled IP packet into an SR policy, it is necessary to create a label stack for that packet, and to 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. If the ENLP sub-TLV appears more than once, the update is considered malformed and the "treat-as-withdraw" strategy of [RFC7606] is applied.

The contents of this sub-TLV are used by the SRPM as described in section 4.1 in [I-D.ietf-spring-segment-routing-policy].

```
0  1  2  3
+---------------+---------------+---------------+---------------+
| Type | Length | Flags | RESERVED |
+---------------+---------------+---------------+---------------+
| ENLP          |
+---------------+
```

Where:
Type: TBD1 (to be assigned by IANA from the registry "BGP Tunnel Encapsulation Attribute sub-TLVs" defined in this document Section 8.3).

Length: 3.

Flags: 1 octet of flags. None are defined at this stage. Flags SHOULD be set to zero on transmission and MUST be ignored on receipt.

RESERVED: 1 octet of reserved bits. SHOULD be unset 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 4 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.
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 policy signaled in this Sub-TLV MAY be overridden by local configuration. The section 4.1 of [I-D.ietf-spring-segment-routing-policy] draft describes the behavior on the headend for handling of explicit null label.
2.4.5. 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 Priority sub-TLV does not have any effect on the BGP bestpath selection or propagation procedures. The contents of this sub-TLV are used by the SRPM as described in section 2.11 in [I-D.ietf-spring-segment-routing-policy].

The Priority sub-TLV is optional and it MUST NOT appear more than once in the SR Policy TLV. If the Priority sub-TLV appears more than once, the update is considered malformed and the "treat-as-withdraw" strategy of [RFC7606] is applied.

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Length      |  Priority     |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:

- **Type**: TBD2 (to be assigned by IANA from the registry "BGP Tunnel Encapsulation Attribute sub-TLVs" defined in this document Section 8.3).
- **Length**: 2.
- **Priority**: a 1-octet value.
- **RESERVED**: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt.

2.4.6. Policy Name Sub-TLV

An operator MAY set the Policy Name sub-TLV to attach a symbolic name to the SR Policy candidate path.

Usage of Policy Name sub-TLV is described in section 2 in [I-D.ietf-spring-segment-routing-policy].

The Policy Name sub-TLV may exceed 255 bytes length due to long policy name. Therefore a 2-octet length is required. According to [I-D.ietf-idr-tunnel-encaps], the first bit of the sub-TLV codepoint defines the size of the length field. Therefore, for the Policy Name
The Policy Name sub-TLV is optional and it MUST NOT appear more than once in the SR Policy TLV. If the Policy Name sub-TLV appears more than once, the update is considered malformed and the "treat-as-withdraw" strategy of [RFC7606] is applied.

The Policy Name 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Length                      |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                        Policy Name                          //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:

- **Type**: TBD3 (to be assigned by IANA from the registry "BGP Tunnel Encapsulation Attribute sub-TLVs" defined in this document Section 8.3).
- **Length**: Variable.
- **RESERVED**: 1 octet of reserved bits. SHOULD be unset 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. Extended Color Community

The Color Extended Community as defined in [I-D.ietf-idr-tunnel-encaps] is used to steer traffic into a policy.

When the Color Extended Community is used for the purpose of steering the traffic into an SR Policy, the RESERVED field (as defined in [I-D.ietf-idr-tunnel-encaps]) is changed as follows:

```
1 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|C O|        RESERVED           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
where CO bits are defined as the "Color-Only" bits. [I-D.ietf-spring-segment-routing-policy] defines the influence of these bits on the automated steering of BGP Payload traffic onto SR Policies.

4. SR Policy Operations

As described in this document, the consumer of an SR Policy NLRI is not the BGP process. The BGP process is in charge of the origination and propagation of the SR Policy NLRI but its installation and use is outside the scope of BGP. The details of SR Policy installation and use can be referred from [I-D.ietf-spring-segment-routing-policy].

4.1. Configuration and Advertisement of SR Policies

Typically, but not limited to, an SR Policy is configured into a controller.

Multiple SR Policy NLRIs may be present with the same <color, endpoint> tuple but with different content when these SR policies are intended to different head-ends.

The distinguisher of each SR Policy NLRI prevents undesired BGP route selection among these SR Policy NLRIs and allow their propagation across route reflectors [RFC4456].

Moreover, one or more route-target SHOULD be attached to the advertisement, where each route-target identifies one or more intended head-ends for the advertised SR policy.

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 case, the NO_ADVERTISE community MUST be attached to the SR Policy update.

4.2. Reception of an SR Policy NLRI

On reception of an SR Policy NLRI, a BGP speaker MUST determine if it’s first acceptable, then it determines if it is usable.

4.2.1. Acceptance of an SR Policy NLRI

When a BGP speaker receives an SR Policy NLRI from a neighbor it has to determine if it’s acceptable. The following applies:

- 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. If a router supporting this document receives an SR policy update with no route-target extended communities and no NO_ADVERTISE community, the update MUST NOT be sent to the SRPM. Furthermore, it SHOULD be considered to be malformed, and the "treat-as-withdraw" strategy of [RFC7606] is applied.

- 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, assigned by IANA (see Section 8) from the "BGP Tunnel Encapsulation Attribute Tunnel Types" registry).

A router that receives an SR Policy update that is not valid according to these criteria MUST treat the update as malformed. The route MUST NOT be passed to the SRPM, and the "treat-as-withdraw" strategy of [RFC7606] is applied.

A unacceptable SR Policy update that has a valid NLRI portion with invalid attribute portion MUST be considered as a withdraw of the SR Policy.

### 4.2.2. Usable SR Policy NLRI

If one or more route-targets are present, then at least one route-target MUST match one of the BGP Identifiers of the receiver in order 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 no one matches any of the local BGP Identifiers, then, while the SR Policy NLRI is acceptable, it is not usable on the receiver node. It has to be noted that if the receiver has been explicitly configured to do so, it MAY propagate the SR Policy NLRI to its neighbors as defined in Section 4.2.4.

The SR Policy candidate paths encoded by the usable SR Policy NLRIs are sent to the SRPM.

### 4.2.3. Passing a usable SR Policy NLRI to the SRPM

Once BGP has determined that the SR Policy NLRI is usable, BGP 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 path-selection process as described in section 2.4 in [I-D.ietf-spring-segment-routing-policy].

The SRPM applies the rules defined in section 2 in [I-D.ietf-spring-segment-routing-policy] to determine whether the SR Policy candidate path is valid and to select the best candidate path among the valid SR Policy candidate paths.

4.2.4. Propagation of an SR Policy

By default, a BGP node receiving an SR Policy NLRI MUST NOT propagate it to any EBGP neighbor.

However, a node MAY be explicitly configured to advertise a received SR Policy NLRI to neighbors according to normal BGP rules (i.e., EBGP propagation by an ASBR or iBGP propagation by a Route-Reflector).

SR Policy NLRIs that have been determined acceptable and valid can be propagated, even the ones that are not usable.

Only SR Policy NLRIs that do not have the NO_ADVERTISE community attached to them can be propagated.

4.3. Flowspec and SR Policies

The SR Policy can be carried in context of a Flowspec NLRI ([RFC5575]). In this case, when the redirect to IP next-hop is specified as in [I-D.ietf-idr-flowspec-redirect-ip], the tunnel to the next-hop is specified by the segment list in the Segment List sub-TLVs. The Segment List (e.g., label stack or IPv6 segment list) is imposed to flows matching the criteria in the Flowspec route to steer them towards the next-hop as specified in the SR Policy SAFI NLRI.

5. Contributors

Kausik Majumdar
Cisco Systems
US

Email: kmajumda@cisco.com

Zafar Ali
Cisco Systems
US

Email: zali@cisco.com
The authors of this document would like to thank Shyam Sethuram, John Scudder, Przemyslaw Krol, Alex Bogdanov, Nandan Saha and Ketan Talaulikar for their comments and review of this document.

7. Implementation Status

Note to RFC Editor: Please remove this section prior to publication, as well as the reference to RFC 7942.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort...
has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Several early implementations exist and will be reported in detail in a forthcoming version of this document. For purposes of early interoperability testing, when no FCFS code point was available, implementations have made use of the following values:

- Preference sub-TLV: 12
- Binding SID sub-TLV: 13
- Segment List sub-TLV: 128

When IANA-assigned values are available, implementations will be updated to use them.

8. IANA Considerations

This document defines new Sub-TLVs in following existing registries:

- Subsequent Address Family Identifiers (SAFI) Parameters
- BGP Tunnel Encapsulation Attribute Tunnel Types
- BGP Tunnel Encapsulation Attribute sub-TLVs

This document also defines following new registries:

- SR Policy List Sub-TLVs
- SR Policy Binding SID Flags
- SR Policy Segment Flags
8.1. Existing Registry: Subsequent Address Family Identifiers (SAFI) Parameters

This document defines a new SAFI in the registry "Subsequent Address Family Identifiers (SAFI) Parameters" that has been assigned by IANA:

<table>
<thead>
<tr>
<th>Codepoint</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>73</td>
<td>SR Policy SAFI</td>
<td>This document</td>
</tr>
</tbody>
</table>

8.2. Existing Registry: BGP Tunnel Encapsulation Attribute Tunnel Types

This document defines a new Tunnel-Type in the registry "BGP Tunnel Encapsulation Attribute Tunnel Types" that has been assigned by IANA:

<table>
<thead>
<tr>
<th>Codepoint</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>SR Policy Type</td>
<td>This document</td>
</tr>
</tbody>
</table>

8.3. Existing Registry: BGP Tunnel Encapsulation Attribute sub-TLVs

This document defines new sub-TLVs in the registry "BGP Tunnel Encapsulation Attribute sub-TLVs" to be assigned by IANA:

<table>
<thead>
<tr>
<th>Codepoint</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Preference sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>13</td>
<td>Binding SID sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>128</td>
<td>Segment List sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>TBD1</td>
<td>ENLP sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>TBD2</td>
<td>Priority sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>TBD3</td>
<td>Policy Name sub-TLV</td>
<td>This document</td>
</tr>
</tbody>
</table>

8.4. New Registry: SR Policy List Sub-TLVs

This document defines a new registry called "SR Policy List Sub-TLVs". The allocation policy of this registry is "First Come First Served (FCFS)" according to [RFC8126].

Following Sub-TLV codepoints are defined:
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MPLS SID sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>2</td>
<td>SRv6 SID sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>3</td>
<td>IPv4 Node and SID sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>4</td>
<td>IPv6 Node and SID for SR-MPLS sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>5</td>
<td>IPv4 Node, index and SID sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>6</td>
<td>IPv4 Local/Remote addresses and SID sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>7</td>
<td>IPv6 Node, index for remote and local pair and SID for SR-MPLS sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>8</td>
<td>IPv6 Local/Remote addresses and SID sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>9</td>
<td>Weight sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>10</td>
<td>IPv6 Node and SID for SRv6 sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>11</td>
<td>IPv6 Node, index for remote and local pair and SID for SRv6 sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>12</td>
<td>IPv6 Local/Remote addresses and SID for SRv6 sub-TLV</td>
<td>This document</td>
</tr>
</tbody>
</table>

8.5. New Registry: SR Policy Binding SID Flags

This document defines a new registry called "SR Policy Binding SID Flags". The allocation policy of this registry is "First Come First Served (FCFS)" according to [RFC8126].

Following Flags are defined:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Specified-BSID-Only Flag (S-Flag)</td>
<td>This document</td>
</tr>
<tr>
<td>1</td>
<td>Drop Upon Invalid Flag (I-Flag)</td>
<td>This document</td>
</tr>
<tr>
<td>2-7</td>
<td>Unassigned</td>
<td></td>
</tr>
</tbody>
</table>

8.6. New Registry: SR Policy Segment Flags

This document defines a new registry called "SR Policy Segment Flags". The allocation policy of this registry is "First Come First Served (FCFS)" according to [RFC8126].

Following Flags are defined:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Segment Verification Flag (V-Flag)</td>
<td>This document</td>
</tr>
<tr>
<td>1</td>
<td>SR Algorithm Flag (A-Flag)</td>
<td>This document</td>
</tr>
<tr>
<td>2-7</td>
<td>Unassigned</td>
<td></td>
</tr>
</tbody>
</table>
9. Security Considerations

TBD.

10. References

10.1. Normative References

[I-D.ietf-idr-tunnel-encaps]

[I-D.ietf-pce-segment-routing]

[I-D.ietf-spring-segment-routing]

[I-D.ietf-spring-segment-routing-policy]


10.2. Informational References


Internet-Draft       Segment Routing Policies in BGP           July 2019

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running
Code: The Implementation Status Section", BCP 205,
RFC 7942, DOI 10.17487/RFC7942, July 2016,

Authors’ Addresses

Stefano Previdi
Individual
IT

Email: stefano@previdi.net

Clarence Filsfils
Cisco Systems, Inc.
Brussels
BE

Email: cfilsfil@cisco.com

Paul Mattes
Microsoft
One Microsoft Way
Redmond, WA 98052
USA

Email: pamattes@microsoft.com

Eric Rosen
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
US

Email: erosen@juniper.net

Dhanendra Jain
Google

Email: dhanendra.ietf@gmail.com
Distribution of Traffic Engineering (TE) Policies and State using BGP-LS

draft-ietf-idr-te-lsp-distribution-12

Abstract

This document describes a mechanism to collect the Traffic Engineering and 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.

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

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."
Internet-Draft  TE Policy State Distribution using BGP-LS   October 2019

This Internet-Draft will expire on April 16, 2020.

Copyright Notice

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

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

Table of Contents

1.  Introduction .............................................. 3
2.  Carrying TE Policy Information in BGP ........................ 5
3.  TE Policy NLRI ........................................... 6
4.  TE Policy Descriptors ..................................... 7
   4.1.  Tunnel Identifier (Tunnel ID) .......................... 8
   4.2.  LSP Identifier (LSP ID) ............................... 8
   4.3.  IPv4/IPv6 Tunnel Head-End Address ...................... 9
   4.4.  IPv4/IPv6 Tunnel Tail-End Address ...................... 9
   4.5.  SR Policy Candidate Path Descriptor .................. 10
   4.6.  Local MPLS Cross Connect ............................. 11
       4.6.1.  MPLS Cross Connect Interface .................... 13
       4.6.2.  MPLS Cross Connect FEC .......................... 14
5.  MPLS-TE Policy State TLV ................................. 15
   5.1.  RSVP Objects ....................................... 16
   5.2.  PCEP Objects ..................................... 17
6.  SR Policy State TLVs .................................. 18
   6.1.  SR Binding SID .................................... 18
   6.2.  SR Candidate Path State ............................ 20
   6.3.  SR Candidate Path Name ............................. 22
   6.4.  SR Candidate Path Constraints ...................... 22
       6.4.1.  SR Affinity Constraint ......................... 24
       6.4.2.  SR SRLG Constraint ............................. 25
       6.4.3.  SR Bandwidth Constraint ......................... 26
       6.4.4.  SR Disjoint Group Constraint ................... 26
   6.5.  SR Segment List ................................... 28
   6.6.  SR Segment ....................................... 31
       6.6.1.  Segment Descriptors ............................ 32
   6.7.  SR Segment List Metric ............................ 39
7.  Procedures ............................................ 41
1. Introduction

In many network environments, traffic engineering (TE) policies are instantiated into various forms:

- MPLS Traffic Engineering Label Switched Paths (TE-LSPs).
- IP based tunnels (IP in IP, GRE, etc).
- Segment Routing (SR) Policies as defined in [I-D.ietf-spring-segment-routing-policy]
- 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, IP tunnels (IPv4 or IPv6), SR Policies, etc.

TE Polices 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 reoptimization. While some extensions are proposed in 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 an TE Policy state collection mechanism complementary to the mechanism defined in [RFC8231].

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 own 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 to both meet the application’s requirements and utilize 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.

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 in order 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
+-------------------+-------------------+-------------------+-------------------+
| NLRI Type         | Total NLRI Length |
|                   |                   |
|                   |                   |
+-------------------+-------------------+
//                    //
+-------------------+-------------------+
|                   |
| Link-State NLRI (variable) |
+-------------------+-------------------+
```

A new "NLRI Type" is defined for TE Policy Information as following:

- NLRI Type: TE Policy NLRI value 5.

The format of this new NLRI type is defined in Section 3 below.
3. TE Policy NLRI

This document defines the new TE Policy NLRI-Type and its format as shown in the following figure:

```
+-------------------+
| Protocol-ID       |
+-------------------+
| Identifier (64 bits) |
+-------------------+
// Headend (Node Descriptors) //
+-------------------+
// TE Policy Descriptors (variable) //
+-------------------+
```

where:

- Protocol-ID field specifies the component that owns the TE Policy state in the advertising node. The following new Protocol-IDs are defined and apply to the TE Policy NLRI:

```
+-------------------+
| Protocol-ID       |
| 8                 |
| 9                 |
| NLRI information source protocol |
| RSVP-TE           |
| Segment Routing   |
+-------------------+
```

- "Identifier" is an 8 octet value as defined in [RFC7752].
- "Headend" consists of a Local Node Descriptor (TLV 256) as defined in [RFC7752].
- "TE Policy Descriptors" consists of one or more of the TLVs listed as below:
The Local Node Descriptor TLV MUST include the following Node Descriptor TLVs:

- BGP Router-ID (TLV 516) [I-D.ietf-idr-bgpls-segment-routing-epe], 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 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:

- Member-ASN (TLV 517) [I-D.ietf-idr-bgpls-segment-routing-epe], 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 new BGP-LS TE Policy NLRI type defined in Section 3.
4.1. Tunnel Identifier (Tunnel ID)

The Tunnel Identifier TLV contains the Tunnel ID defined in [RFC3209] and is used for RSVP-TE protocol TE Policies. It has the following format:

```
  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 (LSP ID)

The LSP Identifier TLV contains the LSP ID defined in [RFC3209] and is used for RSVP-TE protocol TE Policies. It has the following format:

```
  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 for RSVP-TE protocol TE Policies. It has following format:

```
0                   1                   2                   3
+-------------------------------+-------------------------------+
|       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 for RSVP-TE protocol TE Policies. It has following format:

```
0                   1                   2                   3
+-------------------------------+-------------------------------+
|       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 [I-D.ietf-spring-segment-routing-policy] and has the following format:

```
+--------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+-------+
|  0     |  1             |  2             |  3             |
+--------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+-------+
|        +----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+-------+
|        | 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.

```
<table>
<thead>
<tr>
<th>Code Point</th>
<th>Protocol Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PCEP</td>
</tr>
<tr>
<td>2</td>
<td>BGP SR Policy</td>
</tr>
<tr>
<td>3</td>
<td>Local (via CLI, Yang model through NETCONF, gRPC, etc.)</td>
</tr>
</tbody>
</table>
```
Flags: 1 octet field with following bit positions defined. Other bits SHOULD be cleared by originator and MUST be ignored by 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 SHOULD be set to 0 by originator and MUST be ignored by 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 indicates the color of the SR Policy
* Originator ASN : 4 octets to carry the 4 byte encoding of the ASN of the originator. Refer [I-D.ietf-spring-segment-routing-policy] Sec 2.4 for details.
* Originator Address : 4 or 16 octets (as indicated by the flags) to carry the address of the originator. Refer [I-D.ietf-spring-segment-routing-policy] Sec 2.4 for details.
* Discriminator : 4 octets to carry the discriminator of the path. Refer [I-D.ietf-spring-segment-routing-policy] Sec 2.5 for details.

4.6. Local MPLS Cross Connect

The Local MPLS Cross Connect TLV identifies a local MPLS state in the form of incoming label and interface followed by an outgoing label and interface. 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].

The Local MPLS Cross Connect TLV has the following format:
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 a FEC Sub-TLV.

The following sub-TLVs are defined for the Local MPLS Cross Connect TLV:
<table>
<thead>
<tr>
<th>Codepoint</th>
<th>Descriptor TLV</th>
</tr>
</thead>
<tbody>
<tr>
<td>556</td>
<td>MPLS Cross Connect Interface</td>
</tr>
<tr>
<td>557</td>
<td>MPLS Cross Connect FEC</td>
</tr>
</tbody>
</table>

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 address or an IPv6 address.

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             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 to the incoming label.

The MPLS Cross Connect FEC sub-TLV has the following format:

```
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)                       //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                     Prefix (variable)                       //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- **Type**: 557
- **Length**: variable.
- **Flags**: 1 octet of flags defined as follows:

```
0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-
4 |             |
+-+-+-+-+-+-+-+-+-
```

* 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 Policy and it is carried in the optional non-transitive BGP Attribute "LINK_STATE Attribute" defined in [RFC7752]. These MPLS-TE Policy characteristics include the characteristics and attributes of the policy, 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Object-origin | Address Family|            RESERVED           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

//        MPLS-TE Policy State Objects (variable)              //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

where:

MPLS-TE Policy State TLV

- Type: 1200

- Length: the total length of the MPLS-TE Policy State TLV not including 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. Following object-origin codepoints are defined in this document.
Address Family: describes the address family used to setup the
MPLS-TE policy. The following address family values are defined
in this document:

<table>
<thead>
<tr>
<th>Code</th>
<th>Dataplane</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MPLS-IPv4</td>
</tr>
<tr>
<td>2</td>
<td>MPLS-IPv6</td>
</tr>
</tbody>
</table>

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]
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 consists 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**

   Segment Routing Policy (SR Policy) architecture is specified in [I-D.ietf-spring-segment-routing-policy]. A SR Policy can comprise of 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 section defines the various TLVs which enable the headend to report the state of an SR Policy, its CP(s), SID-List(s) and their status. These TLVs are carried in the optional non-transitive BGP Attribute "LINK_STATE Attribute" defined in [RFC7752] and enable the same consistent form of reporting for SR Policy state irrespective of the Protocol-Origin used to provision the policy. Detailed procedure is described in Section 7.

6.1. **SR Binding SID**

   The SR Binding SID (BSID) is an optional TLV that provides the BSID and its attributes for the SR Policy CP. The TLV MAY also optionally contain the Provisioned BSID value for reporting when explicitly provisioned.

   The TLV has the following format:
Internet-Draft  TE Policy State Distribution using BGP-LS  October 2019

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 [I-D.ietf-spring-segment-routing-policy]. Other bits SHOULD be cleared by originator and MUST be ignored by receiver.

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 provisioned 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. SHOULd be set to 0 by originator and MUST be ignored by receiver.

- Binding SID: It indicates the operational or allocated BSID value for the CP based on the status flags.

- Provisioned BSID: It is used to report the explicitly provisioned BSID value regardless of whether it is successfully allocated or not. The field is set to value 0 when BSID has not been specified or provisioned for the CP.

The BSID fields above are 4 octet carrying the MPLS Label or 16 octets 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 SHOULD be set to 0 by originator and MUST be ignored by the receiver.

```
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     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

6.2. SR Candidate Path State

The SR Candidate Path (CP) State TLV provides the operational status and attributes of the SR Policy at the CP level. The TLV has the following format:

```
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
o Length: 8 octets

o Priority: 1 octet value which indicates the priority of the CP. Refer Section 2.12 of [I-D.ietf-spring-segment-routing-policy].

o RESERVED: 1 octet. SHOULD be set to 0 by originator and MUST be ignored by receiver.

o 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 [I-D.ietf-spring-segment-routing-policy]. Other bits SHOULD be cleared by originator and MUST be ignored by 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|           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                |
H H H H H H H H H H H H H H H H
```

where:

* S-Flag: Indicates the CP is in 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 local template when set. Refer Section 8.5 of [I-D.ietf-spring-segment-routing-policy].

* 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 best preference amongst the available CPs. Refer Section 8.2 of [I-D.ietf-spring-segment-routing-policy].

* T-Flag : Indicates the CP has been marked as eligible for use as Transit Policy on the headend when set. Refer Section 8.3 of [I-D.ietf-spring-segment-routing-policy].

  o Preference : 4 octet value which indicates the preference of the CP. Refer Section 2.7 of [I-D.ietf-spring-segment-routing-policy].

6.3. SR Candidate Path Name

The SR Candidate Path Name TLV is an optional TLV that is used to carry the symbolic name associated with the candidate path. The TLV has the following format:

```
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 Symbolic Name (variable)             //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

  o Type: 1203

  o Length: variable

  o CP Name : Symbolic name for the CP. It is a string of printable ASCII characters without a NULL terminator.

6.4. SR Candidate Path Constraints

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 by the headend. The constraints may also be applied to an explicit path where the headend is expected to validate that the path expresses satisfies the specified constraints and the path is to
be invalidated by the headend when the constraints are no longer met (e.g. due to topology changes).

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   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   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 SHOULD be cleared by originator and MUST be ignored by receiver.

```
0                   1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|D|P|U|A|T|                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* **D-Flag**: Indicates that the CP needs to use SRv6 dataplane when set and SR/MPLS dataplane when clear
* **P-Flag**: Indicates that the CP needs to use only protected SIDs when set
* **U-Flag**: Indicates that the CP needs to use only unprotected SIDs when set
* **A-Flag**: Indicates that the CP needs to use the SIDs belonging to the specified SR Algorithm only when set
* T-Flag: Indicates that the CP needs to use the SIDs belonging to the specified topology only when set
  o RESERVED: 2 octet. SHOULD be set to 0 by originator and MUST be ignored by receiver.
  o MTID: Indicates the multi-topology identifier of the IGP topology that is preferred to be used when the path is setup. When the T-flag is set then the path is strictly using the specified topology SIDs only.
  o Algorithm: Indicates the algorithm that is preferred to be used when the path is setup. When the A-flag is set then the path is strictly using the specified algorithm SIDs only.
  o RESERVED: 1 octet. SHOULD be set to 0 by originator and MUST be ignored by receiver.
  o 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.

6.4.1. SR Affinity Constraint

The SR Affinity Constraint sub-TLV is an optional sub-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]. 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               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:
6.4.2. SR SRLG Constraint

The SR SRLG Constraint sub-TLV is an optional sub-TLV that is used to carry the Shared Risk Link Group (SRLG) values [RFC4202] that are to be excluded from the candidate path. The TLV has the following format:

```
+---------------+---------------+---------------+
<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
<th>SRLG Values</th>
</tr>
</thead>
</table>
+---------------+---------------+---------------+
```

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. SHOULD be set to 0 by originator and MUST be ignored by receiver.
- **Exclude-Any EAG**: the bitmask used to represent the affinities to be excluded from the path.
- **Include-Any EAG**: the bitmask used to represent the affinities to be included in the path.
- **Include-All EAG**: the bitmask used to represent the all affinities to be included in the path.
6.4.3. SR Bandwidth Constraint

The SR Bandwidth Constraint sub-TLV is an optional sub-TLV that is used to indicate the desired bandwidth availability that needs to be ensured for the candidate path. 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           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          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.

6.4.4. SR Disjoint Group Constraint

The SR Disjoint Group Constraint sub-TLV is an optional sub-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.

The TLV has the following format:
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 SHOULD be cleared by originator and MUST be ignored by receiver.

  0 1 2 3 4 5 6 7
  ++++++++-----
  |S|N|L|F|I|
  +++++-----

  where:

  - **S-Flag**: Indicates that SRLG disjointness is requested
  - **N-Flag**: Indicates that node disjointness is requested when
  - **L-Flag**: Indicates that link disjointness is requested when
  - **F-Flag**: Indicates that the computation may fallback to a lower level of disjointness amongst the ones requested when all cannot be achieved
  - **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.

- **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 SHOULD be cleared by originator and MUST be ignored by receiver.
where:

* S-Flag : Indicates that SRLG disjointness is achieved
* N-Flag : Indicates that node disjointness is achieved
* L-Flag : Indicates that link disjointness is achieved
* F-Flag : Indicates that the computation has fallen back to a lower level of disjointness that requested.
* I-Flag : Indicates that the computation has fallen back to the best path (e.g. IGP path) and disjointness has not been achieved
* X-Flag : Indicates that the disjointness constraint could not be achieved and hence path has been invalidated

- RESERVED: 2 octets. SHOULD be set to 0 by originator and MUST be ignored by receiver.

- Disjointness Group Identifier : 4 octet value that is the group identifier for a set of disjoint paths

6.5. SR Segment List

The SR Segment List TLV is used to report the SID-List(s) of a candidate path. The TLV has following format:
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 [I-D.ietf-spring-segment-routing-policy]. Other bits SHOULD be cleared by originator and MUST be ignored by receiver.

```
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 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.

  o RESERVED: 2 octet. SHOULD be set to 0 by originator and MUST be ignored by receiver.

  o MTID : 2 octet that indicates the multi-topology identifier of the IGP topology to be used when the T-flag is set.

  o Algorithm: 1 octet that indicates the algorithm of the SIDs used in the SID-List when the A-flag is set.

  o RESERVED: 1 octet. SHOULD be set to 0 by originator and MUST be ignored by receiver.

  o Weight: 4 octet field that indicates the weight associated with the SID-List for weighted load-balancing. Refer Section 2.2 and 2.11 of [I-D.ietf-spring-segment-routing-policy].

  o 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 6.6) 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.
6.6. SR Segment

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 a SR Policy candidate path. It is a sub-TLV of the SR Segment List TLV and has following format:

```
0                   1                   2                   3
+-------------------+-------------------+-------------------+-------------------+
|              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 6.6.1 for details)
- **RESERVED**: 1 octet. SHOULD be set to 0 by originator and MUST be ignored by 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 [I-D.ietf-spring-segment-routing-policy]. Other bits SHOULD be cleared by originator and MUST be ignored by receiver.
* 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 octet 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 SHOULD be set to 0 by originator and MUST be ignored by the receiver.

```
+----------------------------------+
|          Label                        |
+----------------------------------+
```

- Segment Descriptor : variable size Segment descriptor based on the type of segment (refer Section 6.6.1 for details)

- Sub-Sub-TLVs : variable and contains any other optional attributes associated with the specific SID-List.

Currently no Sub-Sub-TLV of the SR Segment sub-TLV is defined.

6.6.1. Segment Descriptors

[I-D.ietf-spring-segment-routing-policy] section 4 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 6.6.
The following types are currently defined:

<table>
<thead>
<tr>
<th>Type</th>
<th>Segment Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Invalid</td>
</tr>
<tr>
<td>1</td>
<td>SR-MPLS Label</td>
</tr>
<tr>
<td>2</td>
<td>SRv6 SID as IPv6 address</td>
</tr>
<tr>
<td>3</td>
<td>SR-MPLS Prefix SID as IPv4 Node Address</td>
</tr>
<tr>
<td>4</td>
<td>SR-MPLS Prefix SID as IPv6 Node Global Address</td>
</tr>
<tr>
<td>5</td>
<td>SR-MPLS Adjacency SID as IPv4 Node Address &amp; Local Interface ID</td>
</tr>
<tr>
<td>6</td>
<td>SR-MPLS Adjacency SID as IPv4 Local &amp; Remote Interface Addresses</td>
</tr>
<tr>
<td>7</td>
<td>SR-MPLS Adjacency SID as pair of IPv6 Global Address &amp; Interface ID for Local &amp; Remote nodes</td>
</tr>
<tr>
<td>8</td>
<td>SR-MPLS Adjacency SID as pair of IPv6 Global Addresses for the Local &amp; Remote Interface</td>
</tr>
<tr>
<td>9</td>
<td>SRv6 END SID as IPv6 Node Global Address</td>
</tr>
<tr>
<td>10</td>
<td>SRv6 END.X SID as pair of IPv6 Global Address &amp; Interface ID for Local &amp; Remote nodes</td>
</tr>
<tr>
<td>11</td>
<td>SRv6 END.X SID as pair of IPv6 Global Addresses for the Local &amp; Remote Interface</td>
</tr>
</tbody>
</table>

6.6.1.1. 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
+-+-+-+-+-+-+-+
|            |
+-+-+-+-+-+-+-+
```

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.

6.6.1.2. 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:
### 6.6.1.3. 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 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. This is valid only when the A-flag has been set in the Segment TLV.

- **IPv4 Node Address**: 4 octet value which carries the IPv4 address associated with the node

### 6.6.1.4. 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 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 address associated with the node
o Algorithm: 1 octet value that indicates the algorithm used for picking the SID

o IPv6 Node Global Address: 16 octet value which carries the IPv6 global address associated with the node

6.6.1.5. Type 5: SR-MPLS Adjacency SID for IPv4 with 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:

```
+-----------------+-+-----------------+-+
| IPv4 Node Address (4 octets) | Local Interface ID (4 octets) |
```

Where:

o IPv4 Node Address: 4 octet value which carries the IPv4 address associated with the node

o Local Interface ID : 4 octet value which carries the local interface ID of the node identified by the Node Address

6.6.1.6. Type 6: SR-MPLS Adjacency SID for IPv4 with 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:

```
+-----------------+-+-----------------+-+
| IPv4 Local Address (4 octets) | IPv4 Remote Address (4 octets) |
```

Where:

o 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).

### 6.6.1.7. Type 7: SR-MPLS Adjacency SID for IPv6 with 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:

```
+---------------------------------------------------------------+
| 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).

### 6.6.1.8. Type 8: SR-MPLS Adjacency SID for IPv6 with 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:
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

6.6.1.9. 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:

```
+-----------------+
| 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

6.6.1.10. Type 10: SRv6 END.X SID as interface 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:

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 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).
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

6.7.  SR Segment List Metric

The SR Segment List Metric sub-TLV describes the metric used for computation of the SID-List. It is used to report the type of metric used in the computation of a dynamic path either on the headend or when the path computation is delegated to a PCE/controller. When the path computation is done on the headend, it is also used to report the calculated metric for the path.

It is a sub-TLV of the SR Segment List TLV and has following format:

```
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 used for path computation. Following metric type codepoints are defined in this document.

<table>
<thead>
<tr>
<th>Code Point</th>
<th>Metric Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>IGP Metric</td>
</tr>
<tr>
<td>1</td>
<td>Min Unidirectional Link Delay [RFC7471]</td>
</tr>
<tr>
<td>2</td>
<td>TE Metric [RFC3630]</td>
</tr>
</tbody>
</table>
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 SHOULD be cleared by originator and MUST be ignored by receiver.

```
0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|M|A|B|V|       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

* M-Flag : Indicates that the metric margin allowed for 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. SHOULD be set to 0 by originator and MUST be ignored by receiver.

metric margin: 4 octets which indicate the metric margin value when 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 octets which indicate the maximum metric value that is allowed when B-flag is set. If the computed path metric crosses the specified bound value then the path is considered as invalid.

metric value: 4 octets which indicate the metric value of the computed path when V-flag is set. This value is available and reported when the computation is successful and a valid path is available.
7. Procedures

The BGP-LS advertisements for the TE Policy NLRI 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 SR Policies, 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 attribute and state is 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 6.2 is included to report the state of the CP. The SR BSID TLV as defined in Section 6.1 is included to report the BSID of the CP when one is either provisioned or allocated by the headend. The constraints for the SR Policy Candidate Path are reported using the SR Candidate Path Constraints TLV as described in Section 6.4. 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 and constraints for the specific SID List.

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 document requires new IANA assigned codepoints.

9.1. BGP-LS NLRI-Types

IANA maintains a registry called "Border Gateway Protocol - Link State (BGP-LS) Parameters" with a sub-registry called "BGP-LS NLRI-Types".

The following codepoints have been assigned by early allocation process by IANA:

+----------------+-----------------+-----------------+
| Type | NLRI Type       | Reference       |
+----------------+-----------------+-----------------+
| 5    | TE Policy NLRI type | this document   |
+----------------+-----------------+-----------------+

9.2. BGP-LS Protocol-IDs

IANA maintains a registry called "Border Gateway Protocol - Link State (BGP-LS) Parameters" with a sub-registry called "BGP-LS Protocol-IDs".

The following Protocol-ID codepoints have been assigned by early allocation process 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 "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 have been assigned by early allocation process by IANA:

<table>
<thead>
<tr>
<th>TLV Code Point</th>
<th>Description</th>
<th>Value defined in</th>
</tr>
</thead>
<tbody>
<tr>
<td>550</td>
<td>Tunnel ID TLV</td>
<td>this document</td>
</tr>
<tr>
<td>551</td>
<td>LSP ID TLV</td>
<td>this document</td>
</tr>
<tr>
<td>552</td>
<td>IPv4/6 Tunnel Head-end address TLV</td>
<td>this document</td>
</tr>
<tr>
<td>553</td>
<td>IPv4/6 Tunnel Tail-end address TLV</td>
<td>this document</td>
</tr>
<tr>
<td>554</td>
<td>SR Policy CP Descriptor TLV</td>
<td>this document</td>
</tr>
<tr>
<td>555</td>
<td>MPLS Local Cross Connect TLV</td>
<td>this document</td>
</tr>
<tr>
<td>556</td>
<td>MPLS Cross Connect Interface TLV</td>
<td>this document</td>
</tr>
<tr>
<td>557</td>
<td>MPLS Cross Connect FEC TLV</td>
<td>this document</td>
</tr>
<tr>
<td>1200</td>
<td>MPLS-TE Policy State TLV</td>
<td>this document</td>
</tr>
<tr>
<td>1201</td>
<td>SR BSID TLV</td>
<td>this document</td>
</tr>
<tr>
<td>1202</td>
<td>SR CP State TLV</td>
<td>this document</td>
</tr>
<tr>
<td>1203</td>
<td>SR CP Name TLV</td>
<td>this document</td>
</tr>
<tr>
<td>1204</td>
<td>SR CP Constraints TLV</td>
<td>this document</td>
</tr>
<tr>
<td>1205</td>
<td>SR Segment List TLV</td>
<td>this document</td>
</tr>
<tr>
<td>1206</td>
<td>SR Segment sub-TLV</td>
<td>this document</td>
</tr>
<tr>
<td>1207</td>
<td>SR Segment List Metric sub-TLV</td>
<td>this document</td>
</tr>
<tr>
<td>1208</td>
<td>SR Affinity Constraint sub-TLV</td>
<td>this document</td>
</tr>
<tr>
<td>1209</td>
<td>SR SRLG Constraint sub-TLV</td>
<td>this document</td>
</tr>
<tr>
<td>1210</td>
<td>SR Bandwidth Constraint sub-TLV</td>
<td>this document</td>
</tr>
<tr>
<td>1211</td>
<td>SR Disjoint Group Constraint sub-TLV</td>
<td>this document</td>
</tr>
</tbody>
</table>

9.4. BGP-LS SR Policy Protocol Origin

This document requests IANA to maintain a new sub-registry under "Border Gateway Protocol - Link State (BGP-LS) Parameters". 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:

<table>
<thead>
<tr>
<th>Code Point</th>
<th>Protocol Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PCEP</td>
</tr>
<tr>
<td>2</td>
<td>BGP SR Policy</td>
</tr>
<tr>
<td>3</td>
<td>Local (via CLI, Yang model through NETCONF, gRPC, etc.)</td>
</tr>
</tbody>
</table>
9.5. BGP-LS TE State Object Origin

This document requests IANA to maintain a new sub-registry under "Border Gateway Protocol - Link State (BGP-LS) Parameters". 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:

<table>
<thead>
<tr>
<th>Code Point</th>
<th>Object Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RSVP-TE</td>
</tr>
<tr>
<td>2</td>
<td>PCEP</td>
</tr>
<tr>
<td>3</td>
<td>Local/Static</td>
</tr>
</tbody>
</table>

9.6. BGP-LS TE State Address Family

This document requests IANA to maintain a new sub-registry under "Border Gateway Protocol - Link State (BGP-LS) Parameters". 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:

<table>
<thead>
<tr>
<th>Code Point</th>
<th>Address Family</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MPLS-IPv4</td>
</tr>
<tr>
<td>2</td>
<td>MPLS-IPv6</td>
</tr>
</tbody>
</table>

9.7. BGP-LS SR Segment Descriptors

This document requests IANA to maintain a new sub-registry under "Border Gateway Protocol - Link State (BGP-LS) Parameters". The new registry is called "SR Segment Descriptor Types" and contains the codepoints allocated to the "Segment Type" field defined in Section 6.6 and described in Section 6.6.1. The registry contains the following codepoints, with initial values, to be assigned by IANA:

<table>
<thead>
<tr>
<th>Code Point</th>
<th>Address Family</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Point</td>
<td>Segment Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>0</td>
<td>Invalid</td>
</tr>
<tr>
<td>1</td>
<td>SR-MPLS Label</td>
</tr>
<tr>
<td>2</td>
<td>SRv6 SID as IPv6 address</td>
</tr>
<tr>
<td>3</td>
<td>SR-MPLS Prefix SID as IPv4 Node Address</td>
</tr>
<tr>
<td>4</td>
<td>SR-MPLS Prefix SID as IPv6 Node Global Address</td>
</tr>
<tr>
<td>5</td>
<td>SR-MPLS Adjacency SID as IPv4 Node Address &amp; Local Interface ID</td>
</tr>
<tr>
<td>6</td>
<td>SR-MPLS Adjacency SID as IPv4 Local &amp; Remote Interface Addresses</td>
</tr>
<tr>
<td>7</td>
<td>SR-MPLS Adjacency SID as pair of IPv6 Global Address &amp; Interface ID for Local &amp; Remote nodes</td>
</tr>
<tr>
<td>8</td>
<td>SR-MPLS Adjacency SID as pair of IPv6 Global Addresses for the Local &amp; Remote Interface</td>
</tr>
<tr>
<td>9</td>
<td>SRv6 END SID as IPv6 Node Global Address</td>
</tr>
<tr>
<td>10</td>
<td>SRv6 END.X SID as pair of IPv6 Global Address &amp; Interface ID for Local &amp; Remote nodes</td>
</tr>
<tr>
<td>11</td>
<td>SRv6 END.X SID as pair of IPv6 Global Addresses for the Local &amp; Remote Interface</td>
</tr>
</tbody>
</table>

9.8. BGP-LS Metric Type

This document requests IANA to maintain a new sub-registry under "Border Gateway Protocol - Link State (BGP-LS) Parameters". The new registry is called "Metric Type" and contains the codepoints allocated to the "metric type" field defined in Section 6.7. The registry contains the following codepoints, with initial values, to be assigned by IANA:

<table>
<thead>
<tr>
<th>Code Point</th>
<th>Metric Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>IGP Metric</td>
</tr>
<tr>
<td>1</td>
<td>Min Unidirectional Link Delay [RFC7471]</td>
</tr>
<tr>
<td>2</td>
<td>TE Metric [RFC3630]</td>
</tr>
</tbody>
</table>

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, and Dhanendra Jain for their review and valuable comments.

13. References

13.1. Normative References

[I-D.ietf-idr-bgpls-segment-routing-epe]

[I-D.ietf-spring-segment-routing-policy]


13.2. Informative References


Authors' Addresses

Stefano Previdi
Email: stefano@previdi.net

Ketan Talaulikar (editor)
Cisco Systems, Inc.
India
Email: ketant@cisco.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
Apstra
Email: jefftant.ietf@gmail.com
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

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
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-isr-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

- the type of calculation to be used (e.g. shortest path)
- the metric type to be used (e.g. IGP metric or TE metric)
- 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

- SR Algorithm TLV to indicate the participation of a node in a flex algorithm computation
- 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:

```
+----------------+----------------+
| Type            | Length         |
+----------------+----------------+
| Flex-Algorithm | Metric-Type    |
| Calc-Type       | Priority       |
|                  | sub-TLVs       |
```

Figure 1: Flex Algorithm Definition TLV

where:

- **Type**: TBD (see IANA Considerations Section 3)
- **Length**: variable. Minimum of 8 octets.
- **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].
- **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].

- **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].

- **Priority**: 1 octet value between 0 and 255 inclusive that specifies the priority of the FAD.

- **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:

- **IS-IS**, as defined by the ISIS Flexible Algorithm Definition sub-TLV in [I-D.ietf-lsr-flex-algo].

- **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
   +-----------------------------------------------+
   |                       Type                  |
   +-----------------------------------------------+
   |                                      Length |
   +-----------------------------------------------+
   |                                      EAG     |
   +-----------------------------------------------+
```

where:
2.2. Flex Algo Include Any Affinity

The Flex Algo Include 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Type                |              Length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Include-Any EAG (variable) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- **Type**: TBD (see IANA Considerations Section 3)
- **Length**: variable, dependent on the size of the Extended Admin Group. MUST be a multiple of 4 octets.
- **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:

```
+--------+--------+--------+--------+
| Type   | Length | Include-All EAG (variable) |
+--------+--------+--------+--------+
```

where:

- **Type**: TBD (see IANA Considerations Section 3)
- **Length**: variable, dependent on the size of the Extended Admin Group. MUST be a multiple of 4 octets.
- **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.
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]


7.2. Informative References

[I-D.ietf-idr-bgp-ls-segment-routing-ext]

[I-D.ietf-spring-segment-routing-policy]


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
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

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 August 26, 2019.

Copyright Notice

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

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

Table of Contents

1. Introduction .................................................. 2
2. Terminology .................................................. 3
3. Problem and Requirement ...................................... 3
4. BGP-LS Extensions for S-BFD Discriminator ................. 4
5. IANA Considerations .......................................... 6
6. Manageability Considerations ............................... 6
   6.1. Operational Considerations ........................... 6
   6.2. Management Considerations ....................... 6
7. Security Considerations ...................................... 6
8. Acknowledgements ............................................ 7
9. References .................................................. 7
   9.1. Normative References ............................. 7
   9.2. Informative References .......................... 7
Authors’ Addresses ........................................... 8

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.

Li, et al. Expires August 26, 2019
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:
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.

- IS-IS, as defined by the S-BFD Discriminators sub-TLV in [RFC7883].
- 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.

<table>
<thead>
<tr>
<th>Code Point</th>
<th>Description</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>S-BFD Discriminators TLV</td>
<td>variable</td>
</tr>
</tbody>
</table>

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...
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


9.2. Informative References
[I-D.ietf-mpls-seamless-mpls]

[I-D.ietf-spring-segment-routing-policy]


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
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 components 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

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 9, 2020.
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)
Bidirectional Path
  Segment List
    Weight
    Path Segment
    Segment
    ...
  Reverse Segment List
    Weight
    Path 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-spring-segment-routing-policy]. 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.
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:

```plaintext
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 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-computation 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):
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:

<table>
<thead>
<tr>
<th>Codepoint</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Path Segment sub-TLV</td>
<td>This document</td>
</tr>
</tbody>
</table>

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]

[I-D.ietf-idr-te-lsp-distribution]
8.2. Informative References

[I-D.gandhi-spring-udp-pm]
Gandhi, R., Filipsils, 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
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-to-end path protection. Path Segment can be also used to correlate two unidirectional 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

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 April 25, 2019.
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. Furthermore, 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 carrying 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 identifying 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 List
    - Weight
    - Path 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 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:

- 0: SR-MPLS Path Segment
- 1: SRv6 Path Segment
- 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)
Bidirectional Path
  Segment List
    Weight
    Path Segment
    Segment
    ...
  Reverse Segment List
    Weight
    Path 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 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:

- An Segment List sub-TLV
- 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  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.

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-idf-segment-routing-te-policy] can apply to this document
directly.

Typically but not limit to, the unidirectional or bidirectional SR
policies carrying path identification information are configured by a
controller.

After configuration, the unidirectional or bidirectional SR policies
carrying path identification information 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
reception.

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.
9.2. Informative References

[I-D.gandhi-spring-udp-pm]

[I-D.ietf-mpls-bfd-directed]

Authors’ Addresses

Li, et al.               Expires April 25, 2019
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.

Copyright Notice

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

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

This document captures the design rationale of the route leak 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]).
Topological 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.
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 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


6.2. Informative References

[draft-dickson-sidr-route-leak-solns]
Dickson, B., "Route Leaks -- Proposed Solutions", IETF Internet Draft (expired), March 2012,

[draft-kunzinger-idrp-ISO10747-01]
Kunzinger, C., "Inter-Domain Routing Protocol (IDRP)", IETF Internet Draft (expired), November 1994,

[Gao]
Gao, L. and J. Rexford, "Stable Internet routing without global coordination", IEEE/ACM Transactions on Networking, December 2001,

[Gao-Rexford]
Freedman, M., "Interdomain Routing Policy", Princeton University COS 461 Lecture Notes; Slides 25-27, Spring 2011,

[Gill]

[Giotsas]

[I-D.ietf-idr-aspath-orf]

[I-D.ietf-idr-bgp-open-policy]


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:
Design Discussion of Route Leaks Solution Methods
draft-sriram-idr-route-leak-solution-discussion-02

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

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 6, 2020.

Copyright Notice

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

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


6.2. Informative References

[draft-dickson-sidr-route-leak-solns]

[draft-kunzinger-idrp-ISO10747-01]


Acknowledgements

The authors wish to thank Jared Mauch, Jeff Haas, Job Snijders, Warren Kumari, Amogh Dhamdhere, Jakob Heitz, Geoff Huston, Ruediger Volk, Sue Hares, 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:

Sriram
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
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.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of
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 analyzes 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.

```
+-----------------+        +-----------------+
|                 |        |                 |
|                ++       |                ++
|         BGP-LS   |       |         BGP-LS |
|                ++       |                ++
|    S1----------S2----B1---------B2----T1---------T2 |
|         N1      ++      |         +++     |         N2   ++
| ++++++        +++++   | ++++++++       ++++++++   |
|    S3----------S4----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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| 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 5.2

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:

<table>
<thead>
<tr>
<th>TLV Code</th>
<th>Description</th>
<th>IS-IS/OSPF TLV</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>Remote AS Number</td>
<td>24/21</td>
<td>[RFC5316]/3.3.1</td>
</tr>
<tr>
<td>TBD</td>
<td>IPv4 Remote ASBR ID</td>
<td>25/22</td>
<td>[RFC5316]/3.3.2</td>
</tr>
<tr>
<td>TBD</td>
<td>IPv6 Remote ASBR ID</td>
<td>26/24</td>
<td>[RFC5316]/3.3.3</td>
</tr>
</tbody>
</table>

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 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Remote ASBR ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
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(Metro-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]

[I-D.ietf-idr-bgppls-segment-routing-epe]

[I-D.ietf-ospf-ospfv3-lsa-extend]

[I-D.ietf-teas-native-ip-scenarios]


Wang & Chen
Expires February 15, 2019

[Page 8]


Authors’ Addresses

Aijun Wang
China Telecom
Beiqijia Town, Changping District
Beijing, Beijing 102209 China

Email: wangaj.bri@chinatelecom.cn
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.

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.

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 not expected that this document will ever become a RFC.
time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

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

Copyright Notice

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

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

Table of Contents

1.  Introduction .............................................. 3
2.  Terminology .............................................. 4
3.  Applicability ............................................. 4
4.  Requirements .............................................. 4
5.  Overview ................................................ 6
6.  UDP Message Header ...................................... 7
7.  Hello Message Format ..................................... 8
8.  Hello Message TLVs ....................................... 10
   8.1.  Accepted ASN List TLV ................................ 10
   8.2.  Peering Address TLV .................................. 11
   8.3.  Local Prefix TLV ..................................... 13
   8.4.  Link Attributes TLV .................................. 14
   8.5.  Neighbor TLV ......................................... 16
   8.6.  Cryptographic Authentication TLV ..................... 18
9.  Neighbor Discovery Procedure ............................ 20
   9.1.  Interface Procedures ............................... 20
   9.2.  Adjacency State Machine ......................... 21
      9.2.1.  Down State ..................................... 21
      9.2.2.  Initial State .................................. 22
      9.2.3.  1-Way State .................................... 22
      9.2.4.  2-Way State .................................... 22
      9.2.5.  Adj-Reject State ................................. 23
      9.2.6.  Adj-OK State .................................... 24
      9.2.7.  Accepted State .................................. 24
   9.3.  Adjacency Route .................................... 25
10. Interactions with Base BGP Protocol ..................... 26
Internet-Draft                                                April 2019

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-1svr-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.

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.

- In DC networks where BGP is used as a hop-by-hop routing protocol [RFC7938].

- 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 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.
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
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 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.

```
+------------------+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          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.
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Type                 |      Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Flags       |   State       |      Reserved                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Neighbor AS number                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Neighboring 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.
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:

- Verify subnet matching between the local and remote interface addresses.

- Verify AS numbers based on local policy as well as against the Allowed ASN TLV when one is being exchanged.

- 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 it 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).
- Authentication methods that are need to be enabled in an environment which is not secure
- Local interfaces over which the specific template needs to be applied for BGP neighbor discovery
- 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 document 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.
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.

<table>
<thead>
<tr>
<th>Value</th>
<th>TLV Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td>This document</td>
</tr>
<tr>
<td>1</td>
<td>Accepted ASN List</td>
<td>This document</td>
</tr>
<tr>
<td>2</td>
<td>Peering Address</td>
<td>This document</td>
</tr>
<tr>
<td>3</td>
<td>Local Prefix</td>
<td>This document</td>
</tr>
<tr>
<td>4</td>
<td>Link Attributes</td>
<td>This document</td>
</tr>
<tr>
<td>5</td>
<td>Neighbor</td>
<td>This document</td>
</tr>
<tr>
<td>6</td>
<td>Cryptographic Authentication</td>
<td>This document</td>
</tr>
<tr>
<td>7-65500</td>
<td>Unassigned</td>
<td>This document</td>
</tr>
<tr>
<td>65501-65534</td>
<td>Experimental</td>
<td>This document</td>
</tr>
<tr>
<td>65535</td>
<td>Reserved</td>
<td>This document</td>
</tr>
</tbody>
</table>

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
16. References

16.1. Normative References


16.2. Informative References


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
Apstra
USA

Email: nikos@apstra.com

Xu, et al. Expires October 18, 2019 [Page 33]
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 specify the extension 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.
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 (TED) 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. PathMTU 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 of LDP tunnels. 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 [I-D.ietf-spring-segment-routing] leverages the source routing paradigm. Segment Routing can be directly applied to the MPLS architecture with no change on the forwarding plane [I-D.ietf-spring-segment-routing-mpls] and applied to the IPv6 architecture with a new type of routing header called the SR header (SRH) [I-D.ietf-6man-segment-routing-header]. [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 than traditional MPLS packet size.

This document specify the extension to BGP Link State (BGP-LS) to carry link maximum transmission unit (MTU) messages.

2. 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 LSDBs and distribute it to a consumer, either directly or via a peer BGP speaker (typically a dedicated Route Reflector). As for how IGP collects link MTU information and stores it in LSDB, which is beyond the scope of this article.

As per [RFC7752], the collection of link-state and TE information and its distribution to consumers is shown in the following figure.

![Figure 1: Collection of Link-State and TE Information](image-url)
3. BGP_LS Extensions for Path 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.

The format of the sub-TLV is as shown below.

```
| TYPE   | TBD |
| LENGTH | Total length of the value field, it should be 2 |
| VALUE  | 2-byte MTU value of the link |
```

Figure 2. Sub-TLV Format for 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.

4. IANA Considerations

This document requests assigning a new code-points from the BGP-LS Link Descriptor and Attribute TLVs registry as specified in sections 3.

5. Security Considerations

This document does not introduce security issues beyond those discussed in RFC7752.

6. Acknowledgements

The authors of this document would like to thank Gang Yan, Peng Wu, Zhenbin Li and Gang Zhao for their comments.
7. References

7.1. Normative References


7.2. Informative References

[I-D.ietf-6man-segment-routing-header]

[I-D.ietf-idr-bgp-ls-segment-routing-ext]

[I-D.ietf-spring-segment-routing]

[I-D.ietf-spring-segment-routing-mpls]


Internet-Draft BGP-LS Extensions for Advertising Path MTU      July 2019

Border Gateway Protocol 4 (BGP-4)", RFC 4271,
DOI 10.17487/RFC4271, January 2006,

S. Ray, "North-Bound Distribution of Link-State and
Traffic Engineering (TE) Information Using BGP", RFC 7752,
DOI 10.17487/RFC7752, March 2016,

Authors’ Addresses

Yongqing Zhu
China Telecom
109, West Zhongshan Road, Tianhe District.
Guangzhou  510000
China

Email: zhuyq@gsta.com

Zhibo Hu
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: huzhibo@huawei.com

Gang Yan
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: yangang@huawei.com

Junda Yao
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: yaojunda@huawei.com